UNIVERSITY OF CAGLIARI

PhD COURSE IN MATHEMATICS AND SCIENTIFIC COMPUTING

Scientific Disciplinary Sector: MAT/08

Applications of low-rank approximation: complex networks and inverse problems

SUPERVISOR:

Prof. Giuseppe Rodriguez

CANDIDATE:

Caterina Fenu

Academic year 2013 – 2014

A matematikus olyan gép, amely a kávét tételekké alakítja
(A mathematician is a machine for turning coffee into theorems)

— Alfréd Rényi (1921 – 1970)

Contents

List of Figures

List of Tables

Abstract

The use of low-rank approximation is crucial when one is interested in solving problems of large dimension. In this case, the matrix with reduced rank can be obtained starting from the singular value decomposition considering only the largest components. This thesis describes how the use of the low-rank approximation can be applied both in the analysis of complex networks and in the solution of inverse problems.

In the first case, it will be explained how to identify the most important nodes or how to determine the ease of traveling between them in large-scale networks that arise in many applications. The use of low-rank approximation is presented both for undirected and directed networks, whose adjacency matrices are symmetric and nonsymmetric, respectively.

As a second application, we propose how to identify inhomogeneities in the ground or the presence of conductive substances. This survey is addressed with the aid of electromagnetic induction measurements taken with a ground conductivity meter. Starting from electromagnetic data collected by this device, the electrical conductivity profile of the soil is reconstructed with the aid of a regularized damped Gauss–Newton method. The inversion method is based on the low-rank approximation of the Jacobian of the function to be inverted.

Acknowledgements

There are many people that I would like to thank for helping me during my Ph.D., both from the professional point of view and the personal one.

First of all my "academic dad" Giuseppe Rodriguez, for his guidance and his support over the past four years. I couldn't have wished for a better advisor. I am also grateful to the entire research group based on Viale Merello, especially Francesco and Luisa, for making me feel at home. I would like also to thank Prof. Lothar Reichel for the lasting collaboration and inspiration.

I am extremely grateful to all the people in my life who encouraged me. First of all my parents and my sister who always trusted in me. My second family, Ale, Fra, Mu, Silvy and Vale, for the (many many!) years of friendship. My Businco family, especially Ale, Cesare, Simona and Matteo, and my Abacus family, Beto, Cozio, DD, Faby, Red and Ste. With you I spent the best moments of my life.

Last, but not least, Vanni. Thank you for your constant therapy and for your friεndship.

Cagliari, 2015 C. F.

Introduction

Numerical Linear Algebra is the part of mathematics that studies algorithms which involve matrix computation. It is common to deal with problems of large dimension that cannot be faced directly and require some kind of approximation. A low-rank approximation of a matrix is a matrix with reduced rank that replaces the original one in the resolution of a problem. The Eckart–Young theorem [40] proves that the best low-rank approximation of a matrix can be obtained starting from the singular value decomposition. There are several applications that make use of this kind of approximation. Moreover, there is an increasing interest in tensor decomposition in last years [34, 61, 83, 135]. In this thesis we will describe how the low-rank approximation can be useful both in the analysis of complex networks and in the solution of inverse problems.

First, we will describe an application in Complex Networks Theory (CNT). This is a branch of Graph Theory which has recently gained much attention. In particular, complex networks are used to model interactions between various entities in real life applications, e.g. in computer science, sociology, economics, genetics, epidemiology; see for example [13, 43, 46, 103]. A graph is a pair of sets $G = (V, E)$, with $|V| = n$ and $|E| = m$. The elements of V are called nodes or vertices and those of E are known as edges or arcs. If the edges can be travelled in both directions the network is said to be undirected, directed otherwise. The adjacency matrix corresponding to an unweighted graph is the matrix $A \in \mathbb{R}^{n\times n}$ such that $A_{ij} = 1$ if there is an edge from node i to node j, and $A_{ij} = 0$ if node i and j are not adjacent. This kind of matrices are binary and, in general, nonsymmetric. One of the main issues in CNT is to find the "most important" nodes within a graph G. To this aim, various indices (or metrics) have been introduced to characterize the importance of a node in terms of connection with the rest of the network. The simplest and most classical ones are the *in-degree* and the *out-degree*, that is, the number of nodes that can reach one node or that can be reached from that node, respectively. These metrics do not give global information on the graph, since they only count the number of neighbors of each node. We will focus on indices that can be computed in terms of matrix functions applied to the adjacency matrix of the graph. We can define a class of indices starting from a matrix function

$$f(A) = \sum_{m=0}^{\infty} c_m A^m, \qquad c_m \geq 0.$$

Since $[A^m]_{ij}$ gives the number of paths of length m starting from the node i and ending at node j, $[f(A)]_{ij}$ is a weighted average of all the paths connecting i to j, and describes the ease of travelling between them. We refer to $[f(A)]_{ii}$ as the f-centrality of node i, and $[f(A)]_{ij}$ as the f-communicability between node i and node j.

In the literature, particular attention has been reserved to the exponential function. In [44, 49], the authors refer to $\left[e^A\right]_{ii}$ as the *subgraph centrality* of node i and to

$\left[e^A\right]_{ij}$ as the *subgraph communicability* between node i and node j, in the case of an undirected graph. Recently, the notion of *hub centrality* and *authority centrality* has been introduced [11] in the case of a directed graph.

Benzi and Boito [10], following the techniques described by Golub and Meurant [58, 59], employed quadrature formulas to find upper and lower bounds of bilinear forms of the kind $\mathbf{u}^T f(A)\mathbf{v}$ (with $\mathbf{u}$ and $\mathbf{v}$ unit vectors) in the case of a symmetric adjacency matrix.

If we assume that $[f(A)]_{ii}$ is a measure of the importance of node i, then we can identify the m most important nodes as the m nodes with the largest centrality. In order to do this using Gauss-type quadrature rules, we may apply this method with $\mathbf{u} = \mathbf{e}_i$, for $i = 1, \dots, n$. Since a complex network is generally very large, this approach may be impractical.

We will describe a new computational method to rank the nodes of both undirected and directed unweighted networks, according to the values of the above matrix functions. The idea is to reduce the cardinality of the set of candidate nodes in order to apply Gauss-type quadrature rules to a smaller number of nodes. The first part of the resulting algorithm, called *hybrid method*, is based on a low-rank approximation of the adjacency matrix. If the network is undirected a partial eigenvalue decomposition is used [51], while if the network is directed we make use of a partial singular value decomposition [5]. We will compare the hybrid algorithm to other computational approaches, on test networks coming from real applications, e.g. in software engineering, bibliometrics and social networks. We will also present a block algorithm to compute the entries of the matrix exponential for both symmetric and nonsymmetric matrices, which is particularly efficient on computers with a hierarchical memory structure. This algorithm is based on block Gauss and anti-Gauss quadrature rules. In the case of a nonsymmetric matrix the block approach is necessary to avoid breakdowns during the computation [50, 99].

As a second application, we will present an inversion procedure in geophysics. In particular, we are interested in recovering the conductivity profile of the soil starting from measured data. To this aim, we will make use of the Electromagnetic induction (EMI) technique. This method has had widespread use in hydrological and hydrogeological characterizations [89, 108, 117], hazardous waste characterization studies [62, 100], precision-agriculture applications [26, 54, 134], archaeological surveys [86, 106, 126], geotechnical investigations [110] and unexploded ordnance (UXO) detection [79, 80]. The use of small measurement systems, with rapid response and easy to integrate into mobile platforms, has been the key factor in the success of EMI techniques for near-surface investigations in these fields, as they allow dense surveying and real-time conductivity mapping over large areas in a cost-effective manner. EMI theory and foundations of measurement systems are described in the applied geophysics literature [101, 121, 132]. The basic instrument, usually called a ground conductivity meter (GCM), contains two small coils, a transmitter and a receiver, whose axes can be aligned either vertically or horizontally with respect to the ground surface. An alternating sinusoidal current in the transmitter produces a primary magnetic field H_P, which induces small eddy currents in the subsurface. These currents, in turn, produce a secondary magnetic field H_S which is measured, together with the primary field, at the receiver. The ratio of the secondary to the primary magnetic fields, recorded as in-phase and quadrature components, is then used, along with the instrumental parameters (height above the ground, frequency, inter-coil spacing, and coil configuration), to estimate electrical properties (conductivity and magnetic susceptibility) of the subsurface.

Traditional GCMs have been designed as profiling instruments for apparent electrical conductivity (defined as the conductivity of a homogeneous half-space that produces the same response as measured above the real earth with the same device) mapping, mainly with subsequent qualitative interpretation. Nevertheless, they can be also used to perform sounding surveys to get quantitative estimates of depth variations in true electrical conductivity. For this purpose, different approaches have been considered. Assuming a linear dependence between the GCM response and the subsurface electrical conductivity, McNeill [101] presented a method to estimate conductivities for simple multi-layered earth models, which is applicable for low induction numbers

$$B = \frac{r}{\delta} = r\sqrt{\frac{\mu_0 \omega \sigma}{2}} \ll 1,$$

under the assumption of uniform electrical conductivity σ. Here r is the inter-coil distance while δ represents the *skin depth* (the depth at which the principal field H_P has been attenuated by a factor e^{-1}); $\mu_0 = 4\pi 10^{-7}$ H/m is the magnetic permeability of free space and $\omega = 2\pi f$, where f is the operating frequency of the device in Hz.

Adopting the same linear model of McNeill [101], Borchers et al. [20] implemented a Tikhonov inverse procedure. Subsequently, to account for high values of the induction number, Hendrickx et al. [74] fitted the technique of Borchers et al. [20] to the nonlinear model described in Ward and Hohmann [132]. Besides, Deidda et al. [35] proposed a least squares inverse procedure to estimate conductivity profiles under the linear model assumption. All these approaches were reliable and useful, as they provided quantitative estimate of depth variation in true electrical conductivity, but they were not very appealing for the practitioners, as long as the use of traditional GCMs prevailed. In fact, collecting the required multiple measurements at several heights above the ground involves time-consuming, laborious, and costly fieldwork.

We will describe a regularized 1D inversion procedure designed to swiftly manage multiple GCM depth responses [36]. It is based on the coupling of the damped Gauss–Newton method with either the truncated singular value decomposition (TSVD) or the truncated generalized singular value decomposition (TGSVD), and it implements an explicit (exact) representation of the Jacobian to solve the nonlinear inverse problem. To illustrate its performance, we first describe the results obtained inverting synthetic data sets generated over three 1D layered models with very high conductivities. In particular, we analyze the influence of some experimental settings, such as number and type of measurements (vertical and horizontal coil configurations), highlighting the different behavior of TSVD and TGSVD, implemented with two different regularization matrices. In addition, we also investigate how to choose the optimal regularization parameter, both when the noise level in the data is known and when it is not. Besides, measuring the execution time for all numerical experiments, in contrast with Schultz and Ruppel [118], we prove that the analytical computation of the Jacobian, combined with the Broyden update, makes the inversion algorithm more then ten times faster than approximating the Jacobian by finite differences. Finally, we present a real case study: using a field data set measured at a site where an independent electrical resistivity tomography (ERT) was also collected, we assess the reliability of the inverse procedure by comparing the inverted conductivity profile to the profile obtained by ERT.

The plan of the thesis is the following:

The first chapter introduces preliminary notions of the numerical linear algebra that will be used in the rest of the thesis.

The second chapter discusses how the low-rank approximation is useful when trying to identify the most important nodes in a network.

The third chapter investigates the application of low-rank approximations in the regularization of an inverse problem in geophysics.

1. Preliminaries

In this chapter we recall some basic concepts in numerical linear algebra that will be used to show how the low-rank approximation can help in the analysis of both complex networks and inverse problems. The first part is dedicated to the connection between Lanczos algorithms and quadrature formulae investigated by Golub and Meurant [58, 59]. These results, and some new ones presented in Subsections 1.5.6 and 1.5.7, will be used in the second chapter. The second part is devoted to the description of some well known results in the inverse problems field that we will use in the third chapter.

1.1 Eigendecomposition, Singular Value Decomposition and Jordan Canonical Form

Definition 1.1.1 (Eigenvalues and eigenvectors). *Given a square matrix $A \in \mathbb{C}^{n\times n}$, an* **eigenvector** *of A is a vector $\boldsymbol{v} \neq \boldsymbol{0} \in \mathbb{C}^n$ such that $A\boldsymbol{v} = \lambda\boldsymbol{v}$, $\lambda \in \mathbb{C}$. The scalar λ is called the* **eigenvalue** *corresponding to the eigenvector $\boldsymbol{v}$.*

By definition, the eigenvectors of a matrix A are the non-trivial solutions of the linear system $(A - \lambda I)\boldsymbol{v} = \boldsymbol{0}$ and therefore the eigenvalues are the solutions of the *characteristic equation* $\det(A - \lambda I) = 0$. For the fundamental theorem of algebra, the characteristic equation has exactly n roots, counted with multiplicity, i.e., every matrix $A \in \mathbb{C}^{n\times n}$ has n, possibly non distinct, eigenvalues.

Definition 1.1.2 (Normal matrix). *A square matrix $A \in \mathbb{C}^{n\times n}$ is normal if $A^*A = AA^*$, where $A^* = \bar{A}^T$ is the conjugate transpose of A.*

Theorem 1.1.1 (Spectral decomposition). *Let $A \in \mathbb{C}^{n\times n}$ be a normal matrix, then there exist a unitary matrix $V \in \mathbb{C}^{n\times n}$ and a diagonal matrix $\Lambda \in \mathbb{C}^{n\times n}$ such that*

$$A = V\Lambda V^* \tag{1.1.1}$$

where Λ contains the eigenvalues of A and the columns of V are the corresponding (properly normalized) eigenvectors. The identity (1.1.1) is called the **Spectral Decomposition** *or* **Eigendecomposition** *of the matrix A.*

Proof. See [85]. □

Definition 1.1.3 (Singular values and singular vectors). *Given a matrix $A \in \mathbb{C}^{m\times n}$, a scalar $\sigma \geq 0$ is a* **singular value** *for A if and only if there exist unit-length vectors $\boldsymbol{u} \in \mathbb{C}^m$ and $\boldsymbol{v} \in \mathbb{C}^n$ such that*

$$A\boldsymbol{v} = \sigma\boldsymbol{u} \quad \text{and} \quad A^*\boldsymbol{u} = \sigma\boldsymbol{v}. \tag{1.1.2}$$

The vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are called **left-singular** *and* **right-singular vectors** *for σ, respectively.*

Multiplying the (1.1.2) by A^* and A, respectively, we obtain

$$A^*A\boldsymbol{v} = \sigma A^*\boldsymbol{u} = \sigma^2\boldsymbol{v} \quad \text{and} \quad AA^*\boldsymbol{u} = \sigma A\boldsymbol{v} = \sigma^2\boldsymbol{u},$$

that is:

- the right-singular vectors of A are eigenvectors of A^*A;
- the left-singular vectors of A are eigenvectors of AA^*;
- the non-zero singular values of A are the square roots of the non-zero eigenvalues of both A^*A and AA^*.
- the number of zero singular values is $p-r$, where $p = \min\{m, n\}$ and $r = \operatorname{rank}(A)$.

Theorem 1.1.2 (Singular value decomposition)**.** *Let $A \in \mathbb{C}^{m\times n}$ be a matrix, then there exist a unitary matrix $U \in \mathbb{C}^{m\times m}$, a diagonal matrix $\Sigma \in \mathbb{C}^{m\times n}$ and a unitary matrix $V \in \mathbb{C}^{n\times n}$ such that*

$$A = U\Sigma V^* \tag{1.1.3}$$

where Σ contains the singular values of A and the columns of U and V are the corresponding (properly normalized) left and right singular vectors, respectively. The identity (1.1.3) is called the **Singular Value Decomposition** *(SVD) of the matrix A.*

Proof. See [60]. □

Definition 1.1.4 (Jordan canonical form)**.** *The* **Jordan form** *of a square matrix $A \in \mathbb{C}^{n\times n}$ is a matrix $J \in \mathbb{C}^{n\times n}$ such that*

$$Z^{-1}AZ = J = \operatorname{diag}(J_1, \dots, J_p), \tag{1.1.4}$$

where

$$J_k = \begin{bmatrix} \lambda_k & 1 & & \\ & \lambda_k & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix} \in \mathbb{C}^{m_k \times m_k}$$

and $Z \in \mathbb{C}^{n\times n}$ is nonsingular. The elements $\lambda_1, \dots, \lambda_p$ are the eigenvalues of the matrix A and $m_1, \dots, m_p$ satisfy $m_1 + \cdots + m_p = n$.

1.2 Low-rank approximation

Generally speaking, computing a low-rank approximation consists of solving an optimization problem, in which the loss function to be minimized measures how a matrix is an appropriate approximation of a given matrix containing the data, subject to the constraint that the approximating matrix has reduced rank.

Definition 1.2.1 (Low-rank approximation)**.** *A low-rank approximation of a matrix $A \in \mathbb{R}^{m\times n}$ is a matrix $A_r \in \mathbb{R}^{m\times n}$ such that*

$$A_r = \min_{\widehat{A}\in\mathbb{R}^{m\times n}} \|A - \widehat{A}\| \quad \textit{subject to} \quad \operatorname{rank}\widehat{A} \le r. \tag{1.2.1}$$

The following theorem shows that the low-rank approximation problem admits analytic solution in terms of the singular value decomposition of the data matrix.

Theorem 1.2.1 (Eckart–Young). *Let $A = U\Sigma V^T \in \mathbb{R}^{m\times n}$ be the singular value decomposition of A e let U, V and Σ partitioned as follows:*

$$U =: \begin{bmatrix} U_1 & U_2 \end{bmatrix}, \quad \Sigma =: \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}, \quad \text{and} \quad V =: \begin{bmatrix} V_1 & V_2 \end{bmatrix}, \tag{1.2.2}$$

where $\Sigma_1 \in \mathbb{R}^{r\times r}$, $U_1 \in \mathbb{R}^{m\times r}$, and $V_1 \in \mathbb{R}^{n\times r}$. Then the rank-r matrix, obtained from the truncated singular value decomposition (TSVD) $\widehat{A}^* = U_1\Sigma_1 V_1^T$, is such that

$$\|A - \widehat{A}^*\| = \min_{\operatorname{rank} \widehat{A} \le r} \|A - \widehat{A}\|.$$

The minimizer $\widehat{A}^*$ is unique if and only if $\sigma_{r+1} \neq \sigma_r$.

Proof. See [40] □

1.3 Orthogonal polynomials

Definition 1.3.1 (Stieltjes integral). *The Riemann–Stieltjes integral of a function $f : \mathbb{R} \to \mathbb{R}$ with respect to a function $\omega : \mathbb{R} \to \mathbb{R}$ is defined as*

$$\int_a^b f(x)\, d\omega(x) = \lim_{\delta(P)\to 0} \sum_{i=0}^{n-1} f(c_i)(\omega(x_{i+1}) - \omega(x_i)), \qquad c_i \in [x_i, x_{i+1}]$$

where $P = \{a = x_0 < x_1 < \cdots < x_n = b\}$ is a partition of the interval $[a, b]$ and

$$\delta(P) = \max_{x_i \in P} |x_{i+1} - x_i|.$$

The two functions f and ω are respectively called the integrand and the integrator.

In the following we will make use also of the notation

$$\int_a^b f(x)w(x)\, dx,$$

where $w(x)$ is a weight function.

Definition 1.3.2 (Inner product). *Let Π be the space of real polynomials. Given two polynomials $p, q \in \Pi$, the inner product with respect to the measure ω is defined as*

$$\langle p, q\rangle = \int_a^b p(x)q(x)\, d\omega(x), \tag{1.3.1}$$

and the the norm of p is given by

$$\|p\|_\omega = \left(\int_a^b p(x)^2\, d\omega(x)\right)^{\frac{1}{2}}$$

The discrete version of inner product (1.3.1) is given by

$$\langle p, q \rangle = \sum_{j=1}^{m} p(x_j)q(x_j)w_j^2. \tag{1.3.2}$$

The values x_j and w_j are called nodes and weights, respectively. The relation between the two inner products is given by the fact that (1.3.2) can be seen as an approximation of (1.3.1) which, conversely, can be looked as a Stieltjes integral with respect to the measure $\omega(x)$ defined as

$$\omega(x) = \begin{cases} 0 & \text{if} \quad x < t_1, \\ \sum_{j=1}^{i} w_j^2 & \text{if} \quad t_i \le x < t_{i+1},\ i = 1, \dots, m-1, \\ \sum_{j=1}^{m} w_j^2 & \text{if} \quad x \ge t_m. \end{cases} \tag{1.3.3}$$

The space $(L^2_w, \langle \cdot, \cdot \rangle)$ of the functions such that

$$\int_a^b f(x)^2 \, d\omega(x) < \infty$$

with the inner product given by (1.3.1) (or (1.3.2)), is a Hilbert Space.

Definition 1.3.3 (Orthogonal polynomials)**.** *A set of polynomials $\{p_i\}$, $i = 1, 2, \dots,$ is said to be orthogonal with respect to inner products* (1.3.1)*–*(1.3.2) *if $\langle p_i, p_j \rangle = 0, \forall i \neq j$. Moreover, the polynomials p_i are orthonormal if $\langle p_i, p_j \rangle = \delta_{ij}$.*

Theorem 1.3.1 (Three-term recurrence)**.** *For orthonormal polynomials, there exist sequences of coefficients α_k and β_k, $k = 1, 2, \dots,$ such that*

$$\begin{aligned} \beta_{k+1}p_{k+1}(x) &= (x - \alpha_{k+1})p_k(x) - \beta_k p_{k-1}(x), \qquad k = 0, 1, \dots, \\ p_{-1} &=: 0, \qquad \textit{and} \qquad p_0 =: 1/\beta_0 \end{aligned} \tag{1.3.4}$$

with $\alpha_{k+1} = \langle xp_k, p_k \rangle$, $k = 0, 1, \dots,$ and β_k computed such that $\|p_k\|_\omega = 1$.

Proof. See [59]. □

If the orthonormal polynomials exist for every k, then we can associate them an infinite symmetric tridiagonal matrix, called the *Jacobi matrix*, defined as

$$J_\infty = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \beta_3 & \\ & & \ddots & \ddots & \ddots \\ & & & & \end{bmatrix}. \tag{1.3.5}$$

We will denote by J_k its leading principal submatrix of order k.

The three-term recurrence (1.3.4) can be written in matrix form as

$$xP_k(x) = J_k P_k(x) + \beta_k p_k(x) \boldsymbol{e}_k, \tag{1.3.6}$$

where $P_k(x) = [p_0(x), p_1(x), \dots, p_{k-1}(x)]^T$ is the vector of the first k orthonormal polynomials valued at x, J_k is the Jacobi matrix of order k and $\boldsymbol{e}_k$ is the last column of the identity matrix of order k.

Theorem 1.3.2. *The zeros $\theta_j^{(k)}$ of the orthonormal polynomial p_k are the eigenvalues of the matrix J_k, called the* **Ritz values**, *and $P_k(\theta_j^{(k)})$ are the corresponding (unnormalized) eigenvectors.*

Proof. The proof follows easily from (1.3.6). □

Following [59] we refer to matrix polynomials as polynomials whose coefficients are square matrices.

Definition 1.3.4. *A matrix polynomial $p(x)$ of order k, $x \in \mathbb{R}$, is defined as*

$$p(x) = \sum_{j=0}^{i} x^j C_j$$

where the coefficients C_j are given square matrices of order k.

Definition 1.3.5 (Inner product)**.** *The inner product of two matrix polynomials p and q is defined as*

$$\langle p, q \rangle = \int_a^b p(x) d\omega(x) q(x)^T, \tag{1.3.7}$$

where the measure $\omega(x)$ is a square matrix of order k.

Definition 1.3.6 (Orthogonal matrix polynomials)**.** *A sequence of matrix polynomials $\{p_\ell\}$, $\ell = 0, 1, \dots,$ is said to be orthonormal if*

$$\langle p_i, p_j \rangle = \delta_{i,j} I_k.$$

for every pair (i, j).

Theorem 1.3.3. *Sequences of orthogonal matrix polynomials satisfy a block three-term recurrence*

$$\begin{aligned} x p_{j-1}(x) &= p_j(x)\Gamma_j + p_{j-1}(x)\Omega_j + p_{j-2}(x)\Gamma_{j-1}^T, \qquad j = 1, 2, \dots, \\ p_0(x) &:= I_k, \quad p_{-1}(x) := O_k, \end{aligned} \tag{1.3.8}$$

where O_k denotes the $k \times k$ zero matrix. For each j, the recursion coefficients Γ_j and Ω_j are $k \times k$ matrices with real entries. Moreover, Ω_j is symmetric and Γ_j can be chosen to be upper triangular. Defining

$$P_N(x) := [p_0(x), \dots, p_{N-1}(x)] \in \mathbb{R}^{k \times kN}, \tag{1.3.9}$$

it follows that

$$x P_N(x) = P_N(x) J_N + p_N(x) \Gamma_N E_N^T, \tag{1.3.10}$$

where

$$J_N := \begin{bmatrix} \Omega_1 & \Gamma_1^T & & & \\ \Gamma_1 & \Omega_2 & \Gamma_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & \Gamma_{N-2} & \Omega_{N-1} & \Gamma_{N-1}^T \\ & & & \Gamma_{N-1} & \Omega_N \end{bmatrix} \in \mathbb{R}^{kN \times kN}. \tag{1.3.11}$$

and $E_i := [\boldsymbol{e}_{(i-1)k+1}, \dots, \boldsymbol{e}_{ik}]$ denotes a "block axis vector" of appropriate size with $k \times k$ blocks, that is, its ith block is I_k and all other blocks vanish. The matrix J_N is symmetric, block-tridiagonal, and has bandwidth $2k + 1$.

1.4 Krylov subspaces and Decomposition algorithms

Definition 1.4.1. *Let A be a symmetric matrix of order n and $\boldsymbol{u}$ a given vector of size n. The* **Krylov subspace** *of order k is the linear subspace spanned by the images of $\boldsymbol{u}$ under the first $k-1$ powers of A, that is*

$$\mathcal{K}_k(A, \boldsymbol{u}) = \mathit{span}\{\boldsymbol{u}, A\boldsymbol{u}, A^2\boldsymbol{u}, \dots, A^{k-1}\boldsymbol{u}\}.$$

The matrix

$$K_k = [\boldsymbol{u}, A\boldsymbol{u}, A^2\boldsymbol{u}, \dots, A^{k-1}\boldsymbol{u}]$$

is called the Krylov matrix.

As k increases, the columns of K_k tend to align with the dominant eigenvector of A, that is, K_k is ill conditioned.

1.4.1 The Symmetric Lanczos algorithm

The **Lanczos algorithm** allows to construct an orthonormal basis for the Krylov subspace. The application of k steps of the Lanczos method to the matrix A with initial vector $\boldsymbol{u}$ yields the decomposition

$$AU_k = U_k T_k + \beta_k \boldsymbol{u}_{k+1} \boldsymbol{e}_k^T, \tag{1.4.1}$$

where the matrix $U_k \in \mathbb{R}^{n \times k}$ has orthonormal columns with $U_k \boldsymbol{e}_1 = \boldsymbol{u}$,

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{bmatrix} \in \mathbb{R}^{k \times k}, \tag{1.4.2}$$

$\boldsymbol{u}_{k+1} \in \mathbb{R}^n$ is a unit vector that is orthogonal to the columns of U_k, β_k is a positive scalar, and $\boldsymbol{e}_j$ denotes the jth axis vector of appropriate dimension. After n steps, the Lanczos algorithm yields the factorization

$$A = U_n T_n U_n^T,$$

that is, the matrix A is similar to a tridiagonal matrix.

The following theorems state the connection between Lanczos algorithm and orthogonal polynomials.

Theorem 1.4.1. *The Lanczos vectors $\boldsymbol{u}_k$ can be represented as polynomial in A applied to the initial vector $\boldsymbol{u}_1 = \boldsymbol{u}$. In particular, $\boldsymbol{u}_{k+1} = p_k(A)\boldsymbol{u}_1$ where*

$$p_k(\lambda) = (-1)^{k-1} \frac{\det(T_k - \lambda I)}{\beta_1 \cdots \beta_{k-1}}, \qquad k \geq 1, \qquad \beta_0 p_0(\lambda) \equiv 1. \tag{1.4.3}$$

Proof. See [59] □

Theorem 1.4.2. *Given the Lanczos vectors $\boldsymbol{u}_k$ there exists a measure $\omega(\lambda)$ such that*

$$\langle \boldsymbol{u}_k, \boldsymbol{u}_\ell \rangle = \langle p_k, p_\ell \rangle = \int_a^b p_k(\lambda) p_\ell(\lambda)\, d\omega(\lambda)$$

where $a \leq \lambda_1$ and $b \geq \lambda_n$. This measure is given by

$$\omega(\lambda) = \begin{cases} 0 & \textit{if} \quad \lambda < \lambda_1, \\ \sum_{j=1}^{i} w_j^2 & \textit{if} \quad \lambda_i \leq \lambda < \lambda_{i+1},\ i = 1, \ldots, m-1, \\ \sum_{j=1}^{m} w_j^2 & \textit{if} \quad \lambda \geq \lambda_n, \end{cases}$$

where $A = Q\Lambda Q^T$ is the eigendecomposition of the matrix A and $\boldsymbol{w} = Q^T\boldsymbol{u}$.

Proof. See [59] □

It is easy to show that the polynomials given by (1.4.3) satisfy a three-term recurrence

$$\beta_{k+1}p_{k+1}(\lambda) = (\lambda - \alpha_{k+1})p_k(\lambda) - \beta_k p_{k-1}(\lambda)$$

with initial condition $p_{-1} =: 0$ and $p_0 =: 1/\beta_0$.

1.4.2 The Arnoldi algorithm

If the matrix A is nonsymmetric, an orthonormal basis $\{\boldsymbol{u}_j\}$ for the Krylov subspace $\mathcal{K}_k(A, \boldsymbol{u})$ can be constructed applying a variant of the Gram–Schmidt orthogonalization process. The Arnoldi algorithm consists of orthogonalizing $A\boldsymbol{u}_j$, with $\boldsymbol{u}_1 = \boldsymbol{u}$, instead of orthogonalizing $A^j\boldsymbol{u}$ against the previous vectors. After k steps, the Arnoldi process yields

$$AU_k = U_kH_k + h_{k+1,k}\boldsymbol{u}_{k+1}\boldsymbol{e}_k^T,$$

where $U_k = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$, H_k is an upper Hessenberg matrix and the vector $\boldsymbol{e}_k$ is the kth column of the identity matrix of order k.

If the matrix A is symmetric, then the matrix H_k is a tridiagonal matrix, that is, the Arnoldi algorithm corresponds to the Lanczos algorithm.

1.4.3 The Nonsymmetric Lanczos algorithm

An alternative way to deal with nonsymmetric matrices is the nonsymmetric Lanczos algorithm which constructs two biorthonormal bases $\{\boldsymbol{u}_j\}$ and $\{\tilde{\boldsymbol{u}}_j\}$ for the Krylov subspaces $\mathcal{K}_k(A, \boldsymbol{u})$ and $\mathcal{K}_k(A^T, \tilde{\boldsymbol{u}})$, respectively, with $\langle \boldsymbol{u}, \tilde{\boldsymbol{u}} \rangle \neq 0$. If

$$J_k = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \tilde{\beta}_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_k \\ & & & \tilde{\beta}_k & \alpha_k \end{bmatrix},$$

$U_k = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$ and $\tilde{U}_k = [\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_k]$, the algorithm yields

$$AU_k = U_kJ_k + \tilde{\beta}_{k+1}\boldsymbol{u}_{k+1}\boldsymbol{e}_k^T,$$
$$A^T\tilde{U}_k = \tilde{U}_kJ_k^T + \beta_{k+1}\tilde{\boldsymbol{u}}_{k+1}\boldsymbol{e}_k^T.$$

The drawback of this algorithm is that it may break down, that is, at some step $\langle \boldsymbol{u}_k, \tilde{\boldsymbol{u}}_k \rangle = 0$. There are two different cases:

- one of both $\boldsymbol{u}_k$ and $\tilde{\boldsymbol{u}}_k$ vanish. In this case an invariant susbspace has been found.
- none of $\boldsymbol{u}_k$ and $\tilde{\boldsymbol{u}}_k$ vanishes. This is known as “serious breakdown” and the computation can be continued only applying some strategies.

1.4.4 The Golub–Kahan Bidiagonalization algorithm

In some applications the decomposition of a matrix $M = AA^T$ is crucial. In this case, since M is symmetric, we could apply the Lanczos algorithm. However, the fact that the matrix M is factorized in such a way can be used. Application of ℓ Golub–Kahan bidiagonalization steps to the matrix A with initial vector $\boldsymbol{u}$ yields the decompositions

$$AP_\ell = Q_{\ell+1}B_{\ell+1,\ell}, \qquad A^TQ_\ell = P_\ell B_\ell^T, \tag{1.4.4}$$

where the matrices $P_\ell \in \mathbb{R}^{n\times\ell}$ and $Q_{\ell+1} \in \mathbb{R}^{n\times(\ell+1)}$ have orthonormal columns, $Q_\ell \in \mathbb{R}^{n\times\ell}$ consists of the first ℓ columns of $Q_{\ell+1}$ and $Q_{\ell+1}\mathbf{e}_1 = \boldsymbol{u}$, the matrix $B_{\ell+1,\ell} = [\beta_{jk}] \in \mathbb{R}^{(\ell+1)\times\ell}$ is lower bidiagonal with leading $\ell\times\ell$ submatrix B_ℓ. All diagonal and subdiagonal entries of $B_{\ell+1,\ell}$ may be assumed to be nonvanishing, otherwise the recursions break down and the discussion simplifies. A detailed discussion on Golub–Kahan bidiagonalization is provided, e.g., by Björck [15].

Combining the equations (1.4.4) gives

$$AA^TQ_\ell = Q_\ell B_\ell B_\ell^T + \beta_{\ell+1,\ell}\beta_{\ell,\ell}\mathbf{q}_{\ell+1}\mathbf{e}_\ell^T, \tag{1.4.5}$$

where $\mathbf{q}_{\ell+1}$ denotes the last column of $Q_{\ell+1}$. The matrix

$$T_\ell = B_\ell B_\ell^T \tag{1.4.6}$$

is symmetric and tridiagonal. Therefore, the expression (1.4.5) is a partial symmetric Lanczos tridiagonalization of the symmetric positive semidefinite matrix AA^T.

1.4.5 The Symmetric Block Lanczos Algorithm

The Block Lanczos algorithms can be used in the special case when we start the iterative method with a block-vector instead of a column-vector, as presented in [58, 59]. Let the matrix $X_1 \in \mathbb{R}^{m\times k}$ have orthonormal columns and define $X_0 := O \in \mathbb{R}^{m\times k}$. The recurrence relations of the symmetric block Lanczos method are given by

$$\begin{aligned} \Omega_j &= X_j^TAX_j, \\ R_j &= AX_j - X_j\Omega_j - X_{j-1}\Gamma_{j-1}^T, \qquad j = 1,\dots,N, \\ X_{j+1}\Gamma_j &= R_j, \end{aligned} \tag{1.4.7}$$

and can be expressed in the form

$$A\,[X_1,\dots,X_N] = [X_1,\dots,X_N]\,J_N + X_{N+1}\Gamma_N E_N^T. \tag{1.4.8}$$

Here $X_{j+1}\Gamma_j = R_j$ is a QR factorization, that is $X_{j+1} \in \mathbb{R}^{m\times k}$ has orthonormal columns and $\Gamma_j \in \mathbb{R}^{k\times k}$ is upper triangular, and

$$J_N := \begin{bmatrix} \Omega_1 & \Gamma_1^T & & & \\ \Gamma_1 & \Omega_2 & \Gamma_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & \Gamma_{N-2} & \Omega_{N-1} & \Gamma_{N-1}^T \\ & & & \Gamma_{N-1} & \Omega_N \end{bmatrix} \in \mathbb{R}^{kN\times kN}. \tag{1.4.9}$$

is a block tridiagonal matrix. The symmetric block Lanczos method is said to *break down* at the jth step if R_j is (numerically) rank deficient. In this case, the computations can be continued by replacing (numerically) linearly dependent columns of X_{j+1} by arbitrary columns that are orthogonal to the ranges of the matrices R_j and $X_1, \ldots, X_j$, and then computing the QR factorization $X_{j+1}\Gamma_j = R_j$. The upper triangular matrix $\Gamma_j \in \mathbb{R}^{k\times k}$ so obtained is necessarily singular. For ease of exposition, we assume that the block Lanczos method does not break down during the first N steps, that is, the step $N+1$ of the recursions (1.4.7) determines the matrices Ω_{N+1}, R_{N+1}, X_{N+2}, and Γ_{N+1}.

In this case, each upper triangular matrix Γ_j, for $1 \le j \le N$, is invertible and may be chosen to have positive diagonal elements.

By construction, the block-vectors $X_i \in \mathbb{R}^{m\times k}$ satisfy

$$X_i^T X_j = \delta_{ij} I_k,$$

and it can be shown by induction that

$$X_{i+1} = \sum_{j=0}^{i} A^j X_1 C_j^{(i)}, \quad 0 \le i \le N,$$

for suitable matrices $C_j^{(i)} \in \mathbb{R}^{k\times k}$. Similarly as Golub and Meurant [58, 59], we consider the sequence of matrix polynomials p_i defined by

$$p_i(\lambda) = \sum_{j=0}^{i} \lambda^j C_j^{(i)}, \quad 0 \le i \le N. \tag{1.4.10}$$

The following result from [58, 59] shows that these polynomials are orthonormal with respect to the bilinear form

$$\mathcal{I}(f,g) := \sum_{i=1}^{m} f^T(\lambda_i)\boldsymbol{\alpha}_i\boldsymbol{\alpha}_i^T g(\lambda_i), \tag{1.4.11}$$

where $\alpha : \mathbb{R} \to \mathbb{R}^{k\times k}$ is a discrete matrix-valued distribution with jumps $\boldsymbol{\alpha}_i\boldsymbol{\alpha}_i^T$ at the eigenvalues λ_i of A.

Theorem 1.4.3. *Let $\mathcal{I}$ be defined by* (1.4.11)*. Then the polynomials* (1.4.10) *satisfy*

$$\mathcal{I}(p_i, p_j) = X_{i+1}^T X_{j+1} = \delta_{ij} I_k, \quad 0 \le i, j \le N.$$

Proof. This result is shown in [58, 59]. A proof of a generalization to the nonsymmetric setting is provided in Theorem 1.4.5. □

One can show that, under the assumptions following (1.4.8), the matrix polynomials satisfy the recurrence relation (1.3.8), which is analogous to (1.4.7).

We now state an important characterization of the eigenvalues and eigenvectors of the matrix J_N in (1.4.9) and (1.4.8). This result is part of [39, Theorem 1.1]. It is is reported in [58, 59].

Theorem 1.4.4. *The eigenvalues of J_N are the zeros of* $\det[p_N(\lambda)]$*. Furthermore, defining P_N by* (1.3.9), the unit (right) eigenvector $\boldsymbol{y}_r^{(N)}$ of J_N corresponding to the eigenvalue $\theta_r^{(N)}$ is given by $\boldsymbol{y}_r^{(N)} = P_N^T(\theta_r^{(N)})\boldsymbol{u}_r^{(N)}$, where $\boldsymbol{u}_r^{(N)}$ consists of the first k components of $\boldsymbol{y}_r^{(N)}$. Moreover, $p_N^T(\theta_r^{(N)})\boldsymbol{u}_r^{(N)} = \mathbf{0}$.

Proof. A proof of this result can be found in [39]. Theorem 1.4.6 below and its proof provide a generalization to the nonsymmetric setting. □

1.4.6 The Nonsymmetric Block Lanczos Algorithm

Assume that the matrices $V_1, W_1 \in \mathbb{R}^{m\times k}$ are used as starting block-vectors and that they satisfy $V_1^T W_1 = I_k$, and let $V_0\Delta_0^T = W_0\Gamma_0^T = 0 \in \mathbb{R}^{m\times k}$. The following recurrence relations described by Bai, Day and Ye [8] determine the first N steps of the nonsymmetric block Lanczos method:

$$\begin{aligned}
\Omega_j &= W_j^T\left(AV_j - V_{j-1}\Delta_{j-1}^T\right),\\
R_j &= AV_j - V_j\Omega_j - V_{j-1}\Delta_{j-1}^T,\\
S_j &= A^T W_j - W_j\Omega_j^T - W_{j-1}\Gamma_{j-1}^T,\\
Q_R R_R &= R_j, \quad Q_S R_S = S_j, \qquad\qquad j = 1,\dots,N,\\
W\Sigma V^T &= Q_S^T Q_R,\\
V_{j+1} &= Q_R V\Sigma^{-1/2}, \quad W_{j+1} = Q_S W\Sigma^{-1/2},\\
\Gamma_j &= \Sigma^{1/2}V^T R_R, \quad \Delta_j = \Sigma^{1/2}W^T R_S.
\end{aligned} \tag{1.4.12}$$

Here $Q_R R_R = R_j$ and $Q_S R_S = S_j$ are QR factorizations, where $Q_R, Q_S \in \mathbb{R}^{m\times k}$ have orthonormal columns and $R_R, R_S \in \mathbb{R}^{k\times k}$ are upper triangular. The factorization $W\Sigma V^T = Q_S^T Q_R$ is a singular value decomposition of the right-hand side matrix. The recursions (1.4.12) can be summarized as

$$\begin{aligned}
A\left[V_1,\dots,V_N\right] &= \left[V_1,\dots,V_N\right]J_N + V_{N+1}\Gamma_N E_N^T,\\
A^T\left[W_1,\dots,W_N\right] &= \left[W_1,\dots,W_N\right]J_N^T + W_{N+1}\Delta_N E_N^T,
\end{aligned} \tag{1.4.13}$$

where

$$J_N := \begin{bmatrix} \Omega_1 & \Delta_1^T & & & \\ \Gamma_1 & \Omega_2 & \Delta_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & \Gamma_{N-2} & \Omega_{N-1} & \Delta_{N-1}^T \\ & & & \Gamma_{N-1} & \Omega_N \end{bmatrix} \in \mathbb{R}^{kN\times kN} \tag{1.4.14}$$

The recursion formulas (1.4.12) provide one of many possible implementations of the nonsymmetric block Lanczos method; see [8] for a discussion on the advantages of this particular implementation. We say that the nonsymmetric block Lanczos method *breaks down* at step j if $S_j^T R_j$ is (numerically) singular. The problem of breakdown is more complicated for the nonsymmetric block Lanczos method than for its symmetric counterpart. While breakdown of the symmetric block Lanczos method can always be remedied by the introduction of one or several new vectors, this is not the case for the nonsymmetric block Lanczos method. Instead, the recursions may have to be terminated. This situation is referred to as *serious breakdown*. A sufficient condition for serious breakdown at step j is that the matrix $S_j^T R_j$ is singular, and both matrices S_j and R_j are of full rank. Bai, Day and Ye [8] provide a thorough discussion on breakdown of the recursions (1.4.12) and show that serious breakdown can be circumvented by restarting the nonsymmetric block Lanczos method after introducing an appropriate additional vector in the initial block-vectors W_1 and V_1 (increasing the block-size by 1).

When no breakdown occurs during the recursions (1.4.12), the matrices Γ_j and Δ_j are nonsingular for $1 \le j \le N$. We may assume that the initial block-vectors W_1 and V_1 have been suitably augmented to avoid breakdown.

By construction, the block-vectors W_i and V_i are biorthogonal, i.e., they satisfy

$$V_i^T W_j = \delta_{ij} I_k.$$

One can show by induction that

$$\left.\begin{aligned} V_{i+1} &= \sum_{j=0}^{i} A^j V_1 C_j^{(i)}, \\ W_{i+1} &= \sum_{j=0}^{i} \left(A^T\right)^j W_1 D_j^{(i)}, \end{aligned}\right\} \quad 0 \le i \le N,$$

for suitably chosen matrices $C_j^{(i)}, D_j^{(i)} \in \mathbb{R}^{k \times k}$.

Define the matrix polynomials p_i and q_i by

$$\left.\begin{aligned} p_i(\lambda) &:= \sum_{j=0}^{i} \lambda^j C_j^{(i)}, \\ q_i(\lambda) &:= \sum_{j=0}^{i} \lambda^j D_j^{(i)}, \end{aligned}\right\} \quad 0 \le i \le N. \tag{1.4.15}$$

We now show that these polynomials are biorthogonal with respect to the bilinear form

$$\mathcal{I}(f,g) := \sum_{i=1}^{m} f^H(\bar{\lambda}_i)\boldsymbol{\alpha}_i \boldsymbol{\beta}_i^H g(\lambda_i), \tag{1.4.16}$$

with

$$\begin{aligned} [\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_m] &= W_1^T Q \in \mathbb{C}^{k \times m}, \\ [\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_m] &= \left(Q^{-1} V_1\right)^H \in \mathbb{C}^{k \times m}, \end{aligned} \tag{1.4.17}$$

where $A = Q\Lambda Q^{-1}$. The following is a generalization of [58, Theorem 4.4] and of Theorem 1.4.3.

Theorem 1.4.5. *Let $\mathcal{I}$ be defined by (1.4.16). Then the polynomials (1.4.15) satisfy*

$$\mathcal{I}(q_i, p_j) = W_{i+1}^T V_{j+1} = \delta_{ij} I_k, \quad 0 \le i, j \le N.$$

Proof. We have for $0 \leq i, j \leq N$ that

$$\begin{aligned}
\delta_{ij} I_k = W_{i+1}^T V_{j+1} &= \left(\sum_{s=0}^{i} \left(A^T\right)^s W_1 D_s^{(i)} \right)^T \left(\sum_{t=0}^{j} A^t V_1 C_t^{(j)} \right) \\
&= \sum_{s=0}^{i} \sum_{t=0}^{j} \left(D_s^{(i)}\right)^T W_1^T A^{s+t} V_1 C_t^{(j)} \\
&= \sum_{s=0}^{i} \sum_{t=0}^{j} \left(D_s^{(i)}\right)^T \mathcal{I}(\lambda^{s+t}) C_t^{(j)} \qquad (1.4.18) \\
&= \sum_{s=0}^{i} \sum_{t=0}^{j} \left(D_s^{(i)}\right)^T \mathcal{I}\left(\bar{\lambda}^s, \lambda^t\right) C_t^{(j)} \qquad (1.4.19) \\
&= \sum_{s=0}^{i} \sum_{t=0}^{j} \mathcal{I}\left(\bar{\lambda}^s D_s^{(i)}, \lambda^t C_t^{(j)}\right) \\
&= \mathcal{I}\left(\sum_{s=0}^{i} \bar{\lambda}^s D_s^{(i)}, \sum_{t=0}^{j} \lambda^t C_t^{(j)} \right) = \mathcal{I}(q_i, p_j),
\end{aligned}$$

where $\mathcal{I}$ in (1.4.18) and (1.4.19) is defined as

$$\mathcal{I} f =: \sum_{i=1}^{m} f(\lambda_i) \boldsymbol{\alpha}_i \boldsymbol{\beta}_i^H \qquad (1.4.20)$$

and (1.4.16), respectively, with $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ defined in (1.4.17). $\square$

One can show that, under the assumptions following (1.4.14), the matrix polynomials p_j and q_i satisfy the recurrence relation

$$\begin{aligned}
\lambda p_{j-1}(\lambda) &= p_j(\lambda)\Gamma_j + p_{j-1}(\lambda)\Omega_j + p_{j-2}(\lambda)\Delta_{j-1}^T, \\
\lambda q_{j-1}(\lambda) &= q_j(\lambda)\Delta_j + q_{j-1}(\lambda)\Omega_j^T + q_{j-2}(\lambda)\Gamma_{j-1}^T, \\
p_0(\lambda) &:= I_k, \quad q_0(\lambda) := I_k, \quad p_{-1}(\lambda) := O_k, \quad q_{-1}(\lambda) := O_k,
\end{aligned} \qquad (1.4.21)$$

with the matrix recursion coefficients Δ_j, Γ_j, and Ω_j defined by the nonsymmetric Lanczos recursions (1.4.12). Letting

$$\begin{aligned}
P_N(\lambda) &:= [p_0(\lambda), \ldots, p_{N-1}(\lambda)] \in \mathbb{R}^{k \times kN}, \\
Q_N(\lambda) &:= [q_0(\lambda), \ldots, q_{N-1}(\lambda)] \in \mathbb{R}^{k \times kN},
\end{aligned} \qquad (1.4.22)$$

the recurrence relations (1.4.21) can be expressed as

$$\begin{aligned}
\lambda P_N(\lambda) &= P_N(\lambda) J_N + p_N(\lambda)\Gamma_N E_N^T, \\
\lambda Q_N(\lambda) &= Q_N(\lambda) J_N^T + q_N(\lambda)\Delta_N E_N^T,
\end{aligned} \qquad (1.4.23)$$

where J_N is defined in (1.4.14).

We are in a position to describe properties of the eigenvalues and eigenvectors of the block tridiagonal matrix J_N in (1.4.14). This extends Theorem 1.4.4 to the nonsymmetric setting.

Theorem 1.4.6. *Let the matrix J_N be defined by* (1.4.14) *and let P_N and Q_N be given by* (1.4.22) *with the polynomials p_i and q_i from* (1.4.15)*. Then the following properties hold:*

(1) *The eigenvalues of J_N are the zeros of both $\det[p_N(\lambda)]$ and $\det[q_N(\lambda)]$.*

(2) *The unit right eigenvector $\boldsymbol{y}_r^{(N)}$ of J_N corresponding to the eigenvalue $\theta_r^{(N)}$ is given by $Q_N^T(\theta_r^{(N)})\boldsymbol{u}_r^{(N)}$, where $\boldsymbol{u}_r^{(N)}$ consists of the first k components of $\boldsymbol{y}_r^{(N)}$. Moreover, $q_N^T(\theta_r^{(N)})\boldsymbol{u}_r^{(N)} = \mathbf{0}$.*

(3) *The unit right eigenvector $\boldsymbol{z}_r^{(N)}$ of J_N^T corresponding to the eigenvalue $\theta_r^{(N)}$ is given by $P_N^T(\theta_r^{(N)})\boldsymbol{v}_r^{(N)}$, where $\boldsymbol{v}_r^{(N)}$ consists of the first k components of $\boldsymbol{z}_r^{(N)}$. Further, $p_N^T(\theta_r^{(N)})\boldsymbol{v}_r^{(N)} = \mathbf{0}$.*

Proof. Our proof is inspired by the proof of [39, Theorem 1.1]. We establish the relation between the zeros of $\det[q_N(\lambda)]$ and the eigenvalues of J_N in (1), as well as (2). The remainder of the proof follows similarly and, therefore, is omitted.

Suppose that $J_N\boldsymbol{y} = \theta\boldsymbol{y}$ for $\boldsymbol{y} \neq \mathbf{0}$, and write $\boldsymbol{y}^T = [\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_N^T]$, where $\boldsymbol{y}_i \in \mathbb{C}^k$ for $1 \leq i \leq N$. Notice that

$$\begin{aligned}
\Omega_1\boldsymbol{y}_1 + \Delta_1^T\boldsymbol{y}_2 &= \theta\boldsymbol{y}_1,\\
&\vdots\\
\Gamma_{i-1}\boldsymbol{y}_{i-1} + \Omega_i\boldsymbol{y}_i + \Delta_i^T\boldsymbol{y}_{i+1} &= \theta\boldsymbol{y}_i,\\
&\vdots\\
\Gamma_{N-1}\boldsymbol{y}_{N-1} + \Omega_N\boldsymbol{y}_N &= \theta\boldsymbol{y}_N.
\end{aligned}$$

Since Ω_i, Γ_i, Δ_i are the matrix recurrence coefficients for the polynomials q_i, we obtain by induction that

$$\begin{aligned}
\boldsymbol{y}_{i+1} &= q_i^T(\theta)\boldsymbol{y}_1, \quad 0 \leq i \leq N-1,\\
\mathbf{0} &= q_N^T(\theta)\boldsymbol{y}_1,
\end{aligned}$$

where we have used that the matrices Δ_i are invertible for $1 \leq i \leq N$. Since $\boldsymbol{y} \neq \mathbf{0}$, it follows that $\boldsymbol{y}_1 \neq \mathbf{0}$ and, therefore, $\det[q_N(\theta)] = 0$. This also establishes (2).

It remains to show that every zero of $\det[q_N(\theta)]$ is an eigenvalue of J_N. Suppose that $\det[q_N(\theta)] = 0$. Then there is a vector $\boldsymbol{u} \in \mathbb{C}^{Nk}\backslash\{\mathbf{0}\}$ such that $\boldsymbol{u}^T q_N(\theta) = \mathbf{0}$. By (1.4.23), this implies that

$$J_N Q_N^T(\theta)\boldsymbol{u} = \theta Q_N^T(\theta)\boldsymbol{u}.$$

Since $q_0(\theta) \equiv I_k$, the vector $Q_N^T(\theta)\boldsymbol{u}$ is nonzero and is, thus, a right eigenvector of J_N associated with the eigenvalue θ. This establishes part (1) regarding $\det[q_N(\theta)]$. $\square$

1.5 Bilinear Forms and Quadrature Rules

There are many equivalent ways of defining a matrix function. We will use the one involving the Jordan canonical form (1.1.4). For a deeper discussion see [77].

Definition 1.5.1 (Matrix functions)**.** *Let f be defined on the spectrum of $A \in \mathbb{C}^{n\times n}$ and let A have the Jordan canonical form* (1.1.4)*. Then*

$$f(A) =: Zf(J)Z^{-1} = Z\,\mathrm{diag}\,(f(J_k))Z^{-1}, \tag{1.5.1}$$

where

$$f(J_k) = \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \dots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{bmatrix}$$

Remark 1.5.1. *If A is diagonalizable, then $A = ZJZ^{-1}$ is the eigendecomposition of the matrix A, that is $J = \mathrm{diag}\,(\lambda_i)$ and Z contain the eigenvalues and the eigenvectors of A, respectively. From Definition* 1.5.1 we have

$$f(A) =: Zf(J)Z^{-1} = Z\,\mathrm{diag}\,(f(\lambda_i))Z^{-1}.$$

Let ω be a measure on the interval $[a, b]$ and f a function whose Stieltjes integral and all the moments exist.

Definition 1.5.2 (Quadrature formula)**.** *A quadrature rule is a relation*

$$\int_a^b f(\lambda)\, d\omega = \sum_{j=1}^{N} w_j f(t_j) + R[f].$$

The values t_j and w_j are the nodes and the weights of the quadrature rule, respectively. The rule is said to be of exact degree d if $R[p] = 0$ for all polynomials p of degree d and there are some polynomials q of degree $d+1$ for which $R[q] = 0$.

Approximating the function f with an interpolating polynomial, we can obtain quadrature rules of degree $N-1$. In this case the quadrature formula is said to be interpolatory.

In this thesis we consider the approximation of a Stieltjes integral by Gauss type quadrature rules. The characteristic of these formulae is that the nodes are the roots of the orthogonal polynomial p_N given by the three-term recursive formula (1.3.4) and the weights are computed in order to have an interpolatory formula.

Definition 1.5.3. *The general formula of a Gauss type quadrature rule is given by*

$$I(f) = \int_a^b f(\lambda)d\omega(\lambda) = \sum_{j=1}^{N} w_j f(t_j) + \sum_{k=1}^{M} v_k f(z_k) + R[f], \tag{1.5.2}$$

where weights $[w_j]_{j=1}^N$, $[v_k]_{k=1}^M$ and nodes $[t_j]_{j=1}^N$ are to be determined, while nodes $[z_k]_{k=1}^M$ are prescribed.

If $M = 0$, this leads to the Gauss rule. If $M = 1$ and $z_1 = a$ or $z_1 = b$, we have the Gauss–Radau rule. If $M = 2$ and $z_1 = a$ and $z_2 = b$, this is the Gauss–Lobatto rule.

The remainder term $R[f]$ cannot generally be explicitly computed. If the measure ω is a positive nondecreasing function and if f is smooth enough, then

$$R[f] = \frac{f^{(2N+M)}(\eta)}{(2N+M)!} \int_a^b \prod_{k=1}^{M}(\lambda - z_k) \left[\prod_{j=1}^{N}(\lambda - t_j)\right]^2 d\omega(\lambda), \qquad a < \eta < b.$$

1.5.1 The Gauss rule

Theorem 1.5.1. *Let J_N be the leading principal submatrix of order N of the matrix (1.3.5). Then the nodes t_j^G of the Gauss quadrature formula are the eigenvalues $\theta_i^{(N)}$ of J_N and the weights w_j^G are the squares of the first components of its normalized eigenvectors $\omega_i^{(N)}$.*

Proof. See [59]. □

Theorem 1.5.2. *Suppose that f is such that $f^{(2n)}(\xi) > 0$, $\forall n$ and $\forall \xi$, $a < \xi < b$, and let*

$$\mathcal{G}_N f = \sum_{i=1}^{N} f(\theta_i^{(N)})\omega_i^{(N)}, \tag{1.5.3}$$

where $\theta_i^{(N)}$ and $\omega_i^{(N)}$ are given by Theorem 1.5.1. Then the Gauss rule is exact for polynomials of degree less than or equal $2N-1$ and

$$\mathcal{G}_N f \leq I(f),$$

that is, the Gauss rule is a lower bound for the Stieltjes integral.

The following theorem states the connection between the Gauss quadrature formula and Lanczos algorithm presented in Subsection 1.4.1.

Theorem 1.5.3. *The Gauss rule (1.5.3) can be written as*

$$\mathcal{G}_N f = \mathbf{e}_1^T f(T_N)\mathbf{e}_1, \tag{1.5.4}$$

where T_N is the tridiagonal matrix in (1.4.2).

Proof. See [58, 59] for a proof. □

1.5.2 The Gauss–Radau rule

The Gauss–Radau rule is given by the general formula (1.5.2) when $M = 1$. In order to compute the rule we need to extend the matrix T_N in such a way that it has the prescribed node as an eigenvalue. Suppose that $z_1 = b$, that is the right end of the integration interval. We wish to construct p_{N+1} such that $p_{N+1}(a) = 0$.

From the recursion (1.3.4) we have

$$0 = \beta_{N+1}p_{N+1}(b) = (b - \alpha_{N+1})p_N(b) - \beta_N p_{N-1}(b)$$

that gives

$$\hat{\alpha}_{N+1} = b - \beta_N \frac{p_{N-1}(b)}{p_N(b)}.$$

Then the tridiagonal matrix

$$\hat{T}_{N+1} = \begin{bmatrix} T_N & \beta_{N+1} \\ \beta_{N+1} & \hat{\alpha}_{N+1} \end{bmatrix} \tag{1.5.5}$$

has the prescribed node b as an eigenvalue. As for the Gauss rule, the nodes are the eigenvalues and the weights are the squares of the first components of the eigenvectors. The remainder term for the Gauss–Radau rules is given by

$$R[f] = \frac{f^{(2N+1)}(\eta)}{(2N+1)!} \int_a^b (\lambda - z_1) \left[\prod_{j=1}^{N}(\lambda - t_j)\right]^2 d\omega(\lambda), \qquad a < \eta < b. \tag{1.5.6}$$

Theorem 1.5.4. *Let f be such that $f^{(2n+1)}(\xi) > 0$, $\forall n$ and $\forall \xi$, $a < \xi < b$, and let*

$$\begin{aligned} \hat{\mathcal{G}}^b_{N+1} f &= \sum_{i=1}^{N} f(t_i^b) w_i^b + f(b) v_1^b, \\ \hat{\mathcal{G}}^a_{N+1} f &= \sum_{i=1}^{N} f(t_i^a) w_i^a + f(a) v_1^a, \end{aligned} \tag{1.5.7}$$

with the nodes and the weights computed prescribing b or a, respectively, as an eigenvalue. Then the Gauss–Radau rule is exact for polynomials of degree less than or equal to $2N$ and

$$\hat{\mathcal{G}}^a_{N+1} f \le I(f) \le \hat{\mathcal{G}}^b_{N+1} f.$$

Proof. The proof follows easily form the reminder formula (1.5.6). □

1.5.3 The Anti–Gauss rule

The anti-Gauss quadrature rules for the approximation of $I(f)$ was introduced by Laurie [87]. The idea is to construct a quadrature rule whose error is equal but of opposite sign to the error of the Gauss rule. What makes the anti-Gauss rule attractive is that it can be applied when no useful information with regard to the sign of the quadrature error can be gleaned from a remainder formula.

The $(N+1)$-point anti-Gauss quadrature rule $\mathcal{H}_{N+1}$ associated with the Gauss rule $\mathcal{G}_N$ is characterized by

$$(I - \mathcal{H}_{N+1})p = -(I - \mathcal{G}_N)p \qquad \forall p \in \mathbb{P}^{2N+1}, \tag{1.5.8}$$

where $\mathbb{P}^{2N+1}$ denotes the set of all polynomials of degree at most $2N+1$ (with scalar coefficients). Thus, when $f \in \mathbb{P}^{2N+1}$, the pair of quadrature rules $\mathcal{G}_N f$ and $\mathcal{H}_{N+1} f$ yield upper and lower bounds for $I(f)$. In fact,

$$\mathcal{G}_N f = \mathcal{H}_{N+1} f = I(f) \qquad \text{for} f \in \mathbb{P}^{2N-1}.$$

For more general functions f, the pair of quadrature rules $\mathcal{G}_N f$ and $\mathcal{H}_{N+1} f$ provide upper and lower bounds for $I(f)$ when the coefficients in an expansion of f in terms of orthonormal polynomials with regard to the measure $d\omega$ decay sufficiently rapidly in magnitude; see [25] and the end of Subsection 1.5.9. This condition is difficult to verify computationally; however, computed examples in [25] show that pairs of Gauss and anti-Gauss quadrature rules indeed yield upper and lower bounds for many integrands that are analytic in a large enough region that contains the interval of integration.

The property (1.5.8) is independent of a remainder formula for Gauss quadrature and gives that the error of the Gauss rule can be estimated as

$$\frac{1}{2}\left(H_{N+1} f - \mathcal{G}_N f\right)$$

Using the anti-Gauss rule, the integral (1.5.2) can be approximated by

$$I(f) \approx \frac{1}{2}\left(H_{N+1} f + \mathcal{G}_N f\right).$$

From (1.5.8) follows that

$$\mathcal{H}_{N+1} p = 2I(p) - \mathcal{G}_N p,$$

for all polynomials p of degree $2N+1$. If we define the functional $\mathcal{I}f = 2I(f) - \mathcal{G}_N f$, then $\mathcal{H}_{N+1}f$ is a Gauss rule with $N+1$ nodes for $\mathcal{I}f$. We can define a sequence of orthogonal polynomials $\tilde{p}_j$, $j = 0, \ldots, N+1$, and a tridiagonal matrix $\tilde{J}_{N+1}$ such that a three-term recursive formula (similar to (1.3.4)) holds. In [25, 87] it has been shown that $\tilde{p}_j = p_j$, $j = 0, \ldots, N$ and that

$$\tilde{J}_{N+1} = \begin{bmatrix} J_N & \sqrt{2}\beta_N \\ \sqrt{2}\beta_N & \alpha_{N+1} \end{bmatrix}.$$

As for the Gauss rule, the $N+1$ nodes are the eigenvalues of $\tilde{J}_{N+1}$ and the weights are the squares of the first components of the eigenvectors and so

$$\mathcal{H}_{N+1}f = \boldsymbol{e}_1^T f(\tilde{J}_{N+1})\boldsymbol{e}_1.$$

Remark 1.5.2. *As pointed out in* [59], $\tilde{J}_{N+1}$ is a low-rank modification of J_{N+1}.

1.5.4 The Symmetric Block Gauss quadrature rule

Consider the computation of an approximation of the integral

$$I(f) = \int f(\lambda)\, d\alpha(\lambda), \tag{1.5.9}$$

where $\alpha : \mathbb{R} \to \mathbb{R}^{k\times k}$ is a discrete matrix-valued distribution with jumps $\boldsymbol{\alpha}_i\boldsymbol{\alpha}_i^T$ at the eigenvalues λ_i of A.

We showed in Subsection 1.4.5 that there is a sequence of polynomials p_j that are orthonormal with respect to a bilinear form defined by $d\alpha$ and have $k \times k$ matrix coefficients. The polynomials satisfy a three-term recurrence relation of the form

$$\begin{aligned} \lambda p_{j-1}(\lambda) &= p_j(\lambda)\Gamma_j + p_{j-1}(\lambda)\Omega_j + p_{j-2}(\lambda)\Gamma_{j-1}^T, \qquad j = 1, 2, \ldots, \\ p_0(\lambda) &:= I_k, \quad p_{-1}(\lambda) := O_k, \end{aligned} \tag{1.5.10}$$

where O_k denotes the $k \times k$ zero matrix. For each j, the recursion coefficients Γ_j and Ω_j are $k \times k$ matrices with real entries. Moreover, Ω_j is symmetric and Γ_j can be chosen to be upper triangular; see Section 1.4.5. The p_j are orthonormal with respect to a matrix-valued bilinear form defined by the measure $d\alpha$; see Theorem 1.4.3. Let

$$\lambda P_N(\lambda) = P_N(\lambda)J_N + p_N(\lambda)\Gamma_N E_N^T,$$

be the recurrence relation given by (1.3.10).

Introduce the spectral factorization $J_N = Y_N\Theta_N Y_N^T$, where

$$Y_N = [\boldsymbol{y}_1^{(N)}, \ldots, \boldsymbol{y}_{kN}^{(N)}] \in \mathbb{R}^{kN\times kN}, \qquad \Theta_N = \operatorname{diag}\left[\theta_1^{(N)}, \ldots, \theta_{kN}^{(N)}\right] \in \mathbb{R}^{kN\times kN},$$

where the matrix Y_N is orthogonal and the eigenvalues are ordered according to $\theta_1^{(N)} \le \cdots \le \theta_{kN}^{(N)}$. Consider the expression

$$\mathcal{G}_N f := \sum_{i=1}^{kN} f(\theta_i^{(N)})\boldsymbol{u}_i^{(N)}\left(\boldsymbol{u}_i^{(N)}\right)^T, \tag{1.5.11}$$

where each vector $\boldsymbol{u}_i^{(N)} \in \mathbb{R}^k$ consists of the first k elements of $\boldsymbol{y}_i^{(N)}$. It is shown in [58, 59] that $\mathcal{G}_N$ is a Gauss quadrature rule with respect to a matrix-valued bilinear form defined by the measure $d\alpha$, i.e.,

$$\mathcal{G}_N f = I(f) \qquad \forall f \in \mathbb{P}^{2N-1};$$

related results are discussed in [119]. An alternative and more concise proof of this result is provided in Subsection 1.5.9. We refer to $\mathcal{G}_N$ as an N-block Gauss quadrature rule associated with a bilinear form determined by the matrix measure $d\alpha$. This quadrature rule allows the matrix representation

$$\begin{aligned}\mathcal{G}_N f &= \sum_{i=1}^{kN} f(\theta_i^{(N)})\boldsymbol{u}_i^{(N)}\left(\boldsymbol{u}_i^{(N)}\right)^T = \left[\boldsymbol{u}_1^{(N)},\dots,\boldsymbol{u}_{kN}^{(N)}\right] f(\Theta_N)\left[\boldsymbol{u}_1^{(N)},\dots,\boldsymbol{u}_{kN}^{(N)}\right]^T \\ &= E_1^T Y_N f(\Theta_N) Y_N^T E_1 = E_1^T f(J_N) E_1, \end{aligned} \tag{1.5.12}$$

which shows that for certain functions f, such as $f(x) = \exp(x)$ and $f(x) = 1/(1-cx)$, where c is a suitable constant, the block Gauss rule $\mathcal{G}_N$ can be evaluated efficiently via the right-hand side of (1.5.12).

It is convenient to extend $\mathcal{G}_N$ to allow matrix-valued functions f and g that can be represented by a series with $k \times k$ matrix coefficients with real entries. For later convenience we define the quantity

$$\mathcal{G}_N(f,g) := \sum_{i=1}^{kN} f^T(\theta_i^{(N)})\boldsymbol{u}_i^{(N)}\left(\boldsymbol{u}_i^{(N)}\right)^T g(\theta_i^{(N)}). \tag{1.5.13}$$

1.5.5 The Nonsymmetric Block Gauss quadrature rule

Assume that the matrix $A \in \mathbb{R}^{m\times m}$ is diagonalizable and introduce the spectral factorization $A = Q\Lambda Q^{-1}$, where $Q \in \mathbb{C}^{m\times m}$ is nonsingular and $\Lambda = \operatorname{diag}[\lambda_1,\dots,\lambda_m]$. When $A \neq A^T$, the eigenvalues λ_i may be complex-valued.

We showed in Subsection 1.4.6 that there are two sequences of polynomials p_j and q_j, $j = 0, 1, \dots$, with $k \times k$ matrix coefficients, that are biorthogonal with respect to a bilinear form determined by the matrix-valued function $\mathcal{I}$ in (1.4.20) and satisfy recurrence relations

$$\begin{aligned}\lambda P_N(\lambda) &= P_N(\lambda)J_N + p_N(\lambda)\Gamma_N E_N^T,\\ \lambda Q_N(\lambda) &= Q_N(\lambda)J_N^T + q_N(\lambda)\Delta_N E_N^T,\end{aligned} \tag{1.5.14}$$

where J_N is a block-tridiagonal matrix determined by N steps of the nonsymmetric block Lanczos method described in [8] and discussed in Subsection 1.4.6. We remark that the polynomials p_j and q_j are considered for theoretical purposes only; they are never explicitly stored or utilized in the computations.

We assume that J_N is diagonalizable and write $J_N = Y_N \Theta_N Y_N^{-1}$, where

$$Y_N = [\boldsymbol{y}_1^{(N)},\dots,\boldsymbol{y}_{kN}^{(N)}] \in \mathbb{C}^{kN\times kN}, \quad \Theta_N = \operatorname{diag}\left[\theta_1^{(N)},\dots,\theta_{kN}^{(N)}\right] \in \mathbb{C}^{kN\times kN}.$$

Letting $Z_N := [\boldsymbol{z}_1^{(N)},\dots,\boldsymbol{z}_{kN}^{(N)}] = Y_N^{-H}$, we obtain

$$J_N = Y_N\Theta_N Z_N^H, \qquad J_N^T Z_N = Z_N\bar{\Theta}_N,$$

where the bar denotes complex conjugation. Since the matrices A, V, W have real entries only, so does J_N and, therefore, $J^T = J^H$.

Consider the quadrature rule

$$\mathcal{G}_N f := \sum_{i=1}^{kN} f(\theta_i^{(N)})\boldsymbol{u}_i^{(N)}\left(\boldsymbol{v}_i^{(N)}\right)^H \tag{1.5.15}$$

with respect to a bilinear form determined by $\mathcal{I}$ defined by (1.4.20). Each vector $\boldsymbol{u}_i^{(N)} \in \mathbb{C}^k$ consists of the first k elements of the (right) eigenvector $\boldsymbol{y}_i^{(N)}$ of J_N, and each vector $\boldsymbol{v}_i^{(N)} \in \mathbb{C}^k$ is made up of the first k elements of the (right) eigenvector $\boldsymbol{z}_i^{(N)}$ of J_N^T. We will show in Subsection 1.5.9 that

$$\mathcal{G}_N f = \mathcal{I} f, \qquad \forall f \in \mathbb{P}^{2N-1}.$$

For this reason, we refer to $\mathcal{G}_N$ as an N-block nonsymmetric Gauss quadrature rule associated with $\mathcal{I}$. As for the symmetric case considered above, this quadrature rule can be expressed as

$$\mathcal{G}_N f = E_1^T f(J_N) E_1, \tag{1.5.16}$$

where the matrix J_N is given by (1.4.14).

As above, we extend $\mathcal{I}$ and $\mathcal{G}_N$ to allow matrix-valued functions f and g that can be represented by a series with $k \times k$ matrix coefficients. Thus, we define

$$\mathcal{I}(f,g) \quad := \quad \sum_{i=1}^{m} f^H(\bar{\lambda}_i)\boldsymbol{\alpha}_i\boldsymbol{\beta}_i^H g(\lambda_i), \tag{1.5.17}$$

$$\mathcal{G}_N(f,g) \quad := \quad \sum_{i=1}^{kN} f^H(\bar{\theta}_i^{(N)})\boldsymbol{u}_i^{(N)} \left(\boldsymbol{v}_i^{(N)}\right)^H g(\theta_i^{(N)}). \tag{1.5.18}$$

1.5.6 The Symmetric Block Anti-Gauss quadrature rule

Proceeding similarly as Laurie [87] for the case of a real-valued positive measure (cf. (1.5.8)) we define the $(N+1)$-block anti-Gauss quadrature rule $\mathcal{H}_{N+1}$ to be an $(N+1)$-block quadrature rule such that

$$(I - \mathcal{H}_{N+1})\, f = -\,(I - \mathcal{G}_N)\, f, \qquad f \in \mathbb{P}^{2N+1}. \tag{1.5.19}$$

Since (1.5.19) implies that

$$\mathcal{H}_{N+1} f = (2I - \mathcal{G}_N)\, f, \qquad f \in \mathbb{P}^{2N+1}, \tag{1.5.20}$$

it follows that $\mathcal{H}_{N+1}$ is the (ordinary) $(N+1)$-block Gauss quadrature rule with respect to the bilinear form determined by the matrix-valued function $2\mathcal{I} - \mathcal{G}_N$. We note that the average rule

$$\mathcal{A}_{N+1} := \frac{1}{2}\left(\mathcal{H}_{N+1} + \mathcal{G}_N\right) \tag{1.5.21}$$

is exact for all polynomials of degree up to and including $2N+1$.

Similarly as above, there is a sequence of orthonormal polynomials $\tilde{p}_j$, with $k \times k$ matrix coefficients, such that

$$\begin{aligned} \lambda\tilde{p}_{j-1}(\lambda) &= \tilde{p}_j(\lambda)\tilde{\Gamma}_j + \tilde{p}_{j-1}(\lambda)\tilde{\Omega}_j + \tilde{p}_{j-2}(\lambda)\tilde{\Gamma}_{j-1}^T, \quad j = 1,2,\ldots, \\ \tilde{p}_0(\lambda) &:= I_k, \quad \tilde{p}_{-1}(\lambda) := O_k, \end{aligned} \tag{1.5.22}$$

where the orthonormality is with respect to a bilinear form defined by the matrix-valued measure induced by the function $2\mathcal{I} - \mathcal{G}_N$.

We will show how to determine the symmetric block tridiagonal matrix

$$\tilde{J}_{N+1} = \begin{bmatrix} \tilde{\Omega}_1 & \tilde{\Gamma}_1^T & & & \\ \tilde{\Gamma}_1 & \tilde{\Omega}_2 & \tilde{\Gamma}_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & \tilde{\Gamma}_{N-1} & \tilde{\Omega}_N & \tilde{\Gamma}_N^T \\ & & & \tilde{\Gamma}_N & \tilde{\Omega}_{N+1} \end{bmatrix} \in \mathbb{R}^{k(N+1)\times k(N+1)} \tag{1.5.23}$$

associated with the anti-Gauss rule $\mathcal{H}_{N+1}$ with almost no work from the matrix J_{N+1} related to the $(N+1)$-block Gauss quadrature rule $\mathcal{G}_{N+1}$ defined by a bilinear form determined by the matrix measure $d\alpha$. Analogously to (1.5.12), the $(N+1)$-block anti-Gauss quadrature rule (1.5.20) allows the matrix representation

$$\mathcal{H}_{N+1}f = E_1^T f(\tilde{J}_{N+1})E_1. \tag{1.5.24}$$

The matrix $\tilde{J}_{N+1}$, defined by (1.5.23) and associated with the $(N+1)$-block anti-Gauss quadrature rule (1.5.20), can be obtained, with almost no work, from the matrix J_{N+1} defined by (1.3.11), with N replaced by $N+1$, associated with the $(N+1)$-block Gauss quadrature rule determined by (1.5.11) with N replaced by $N+1$.

It follows from (1.5.10) that the coefficient matrices Ω_i and Γ_i associated with the block Gauss rule (1.5.11), with N replaced by $N+1$, are given by

$$\Omega_i = \mathcal{I}(p_{i-1}, \lambda p_{i-1}), \qquad \Gamma_i = \mathcal{I}(p_i, \lambda p_{i-1}).$$

Similarly, we obtain from (1.5.22) that the coefficients $\tilde{\Omega}_i$ and $\tilde{\Gamma}_i$ associated with the block anti-Gauss rule (1.5.20) satisfy

$$\tilde{\Omega}_i = (2\mathcal{I} - \mathcal{G}_N)\left(\tilde{p}_{i-1}, \lambda\tilde{p}_{i-1}\right), \qquad \tilde{\Gamma}_i = (2\mathcal{I} - \mathcal{G}_N)\left(\tilde{p}_i, \lambda\tilde{p}_{i-1}\right).$$

Therefore, the recursions (1.5.10) and (1.5.22), together with (1.5.20) and Corollary 1.5.1, imply that

$$\begin{array}{rcll} \tilde{\Omega}_i &=& \Omega_i, & 1 \le i \le N, \\ \tilde{\Gamma}_i &=& \Gamma_i, & 1 \le i \le N-1, \\ \tilde{p}_i &=& p_i, & 0 \le i \le N-1. \end{array}$$

It follows that

$$\begin{aligned} \tilde{p}_N\tilde{\Gamma}_N &= \lambda I_k \tilde{p}_{N-1} - \tilde{p}_{N-1}\tilde{\Omega}_N - \tilde{p}_{N-2}\tilde{\Gamma}_{N-1}^T \\ &= \lambda I_k p_{N-1} - p_{N-1}\Omega_N - p_{N-2}\Gamma_{N-1}^T = p_N\Gamma_N. \end{aligned} \tag{1.5.25}$$

Hence,

$$\begin{array}{rcl} \tilde{\Gamma}_N &=& (2\mathcal{I} - \mathcal{G}_N)\left(\tilde{p}_N, \lambda\tilde{p}_{N-1}\right) = 2\mathcal{I}\left(\tilde{p}_N, \lambda\tilde{p}_{N-1}\right) - \mathcal{G}_N\left(\tilde{p}_N, \lambda\tilde{p}_{N-1}\right) \\ &=& 2\left(\Gamma_N\tilde{\Gamma}_N^{-1}\right)^T \mathcal{I}\left(p_N, \lambda p_{N-1}\right) = 2\left(\Gamma_N\tilde{\Gamma}_N^{-1}\right)^T \Gamma_N, \end{array}$$

where $\mathcal{G}_N\left(\tilde{p}_N, \lambda\tilde{p}_{N-1}\right) = \mathbf{0}$, because in view of Theorem 1.4.4, we have

$$\tilde{p}_N^T(\theta_r^{(N)})\boldsymbol{u}_r^{(N)} = \left(\Gamma_N\tilde{\Gamma}_N^{-1}\right)^T p_N^T(\theta_r^{(N)})\boldsymbol{u}_r^{(N)} = \mathbf{0}, \quad 1 \le r \le kN.$$

We conclude that $\tilde{\Gamma}_N^T\tilde{\Gamma}_N = 2\Gamma_N^T\Gamma_N$. Recall that the matrices Γ_N and $\tilde{\Gamma}_N$ are assumed to be invertible, and are chosen to have positive diagonal entries. Therefore,

$$\tilde{\Gamma}_N = \sqrt{2}\Gamma_N, \tag{1.5.26}$$

because the symmetric positive definite matrix $2\Gamma_N^T\Gamma_N$ has a unique Cholesky factorization C^TC with an upper-triangular factor C, whose diagonal is strictly positive.

We turn to the entry $\tilde{\Omega}_{N+1}$. It follows from (1.5.26) that $\Gamma_N\tilde{\Gamma}_N^{-1} = \left(1/\sqrt{2}\right)I_k$, which, in view of (1.5.25), implies that $\tilde{p}_N = \left(1/\sqrt{2}\right)p_N$. Therefore,

$$\tilde{\Omega}_{N+1} = (2\mathcal{I} - \mathcal{G}_N)\left(\tilde{p}_N, \lambda\tilde{p}_N\right) = \mathcal{I}\left(p_N, \lambda p_N\right) = \Omega_{N+1}.$$

In conclusion, the matrix $\tilde{J}_{N+1}$ associated with the $(N+1)$-block anti-Gauss rule can be obtained from the matrix J_{N+1} associated with the $(N+1)$-block Gauss rule by multiplying Γ_N by $\sqrt{2}$.

1.5.7 The Nonsymmetric Block Anti-Gauss quadrature rule

Similarly as in the symmetric case, we seek to determine a matrix-valued $(N+1)$-block anti-Gauss quadrature rule $\mathcal{H}_{N+1}$ such that

$$(\mathcal{I}-\mathcal{H}_{N+1})\,f = -\,(\mathcal{I}-\mathcal{G}_N)\,f, \qquad f\in\mathbb{P}^{2N+1}. \tag{1.5.27}$$

This relation implies, analogously to the discussion following (1.5.19), that $\mathcal{H}_{N+1}$ is an $(N+1)$-block Gauss quadrature rule with respect to a bilinear form determined by the matrix-valued function $2\mathcal{I}-\mathcal{G}_N$. The average rule (1.5.21) with $\mathcal{G}_N$ and $\mathcal{H}_{N+1}$ defined by (1.5.15) and (1.5.27), respectively, is exact for all $p\in\mathbb{P}^{2N+1}$.

Analogously to the discussion above, there are sequences of polynomials $\tilde{p}_j$ and $\tilde{q}_j$, $j=0,1,\ldots$, with real $k\times k$ matrix coefficients, that are biorthogonal with respect to a bilinear form determined by the matrix-valued function $2\mathcal{I}-\mathcal{G}_N$ and satisfy recurrence relations of the form

$$\begin{aligned}
\lambda\tilde{p}_{j-1}(\lambda) &= \tilde{p}_j(\lambda)\tilde{\Gamma}_j+\tilde{p}_{j-1}(\lambda)\tilde{\Omega}_j+\tilde{p}_{j-2}(\lambda)\tilde{\Delta}_{j-1}^T,\\
\lambda\tilde{q}_{j-1}(\lambda) &= \tilde{q}_j(\lambda)\tilde{\Delta}_j+\tilde{q}_{j-1}(\lambda)\tilde{\Omega}_j^T+\tilde{q}_{j-2}(\lambda)\tilde{\Gamma}_{j-1}^T,\\
\tilde{p}_0(\lambda) &:= I_k,\quad \tilde{q}_0(\lambda):=I_k,\quad \tilde{p}_{-1}(\lambda):=O_k,\quad \tilde{q}_{-1}(\lambda):=O_k,
\end{aligned} \tag{1.5.28}$$

for $j=1,2,\ldots$.

We will show how to determine the associated matrix of matrix recursion coefficients

$$\tilde{J}_{N+1}=\begin{bmatrix}
\tilde{\Omega}_1 & \tilde{\Delta}_1^T & & & \\
\tilde{\Gamma}_1 & \tilde{\Omega}_2 & \tilde{\Delta}_2^T & & \\
 & \ddots & \ddots & \ddots & \\
 & & \tilde{\Gamma}_{N-1} & \tilde{\Omega}_N & \tilde{\Delta}_N^T\\
 & & & \tilde{\Gamma}_N & \tilde{\Omega}_{N+1}
\end{bmatrix}\in\mathbb{R}^{k(N+1)\times k(N+1)}, \tag{1.5.29}$$

with almost no work, from the matrix (1.4.14) with N replaced by $N+1$. The $(N+1)$-block nonsymmetric anti-Gauss rule allows the matrix representation

$$\mathcal{H}_{N+1}f=E_1^Tf(\tilde{J}_{N+1})E_1, \tag{1.5.30}$$

analogous to (1.5.24).

As in the symmetric case, the matrix $\tilde{J}_{N+1}$ given by (1.5.29) and associated with the nonsymmetric $(N+1)$-block anti-Gauss rule defined by (1.5.27)) can be determined, with almost no work, from the matrix J_{N+1}, given by (1.4.14) with N replaced by $N+1$, associated with the $(N+1)$-block nonsymmetric Gauss rule (1.5.15) with N replaced by $N+1$.

We obtain from (1.4.21) and (1.5.28) that the coefficients Ω_i, Γ_i, and Δ_i associated with nonsymmetric block Gauss rules and the coefficients $\tilde{\Omega}_i$, $\tilde{\Gamma}_i$, and $\tilde{\Delta}_i$ of nonsymmetric block anti-Gauss rules are given by

$$\begin{aligned}
\Omega_i &= \mathcal{I}(q_{i-1},\lambda p_{i-1}), & \tilde{\Omega}_i &= (2\mathcal{I}-\mathcal{G}_N)\,(\tilde{q}_{i-1},\lambda\tilde{p}_{i-1}),\\
\Gamma_i &= \mathcal{I}(q_i,\lambda p_{i-1}), & \tilde{\Gamma}_i &= (2\mathcal{I}-\mathcal{G}_N)\,(\tilde{q}_i,\lambda\tilde{p}_{i-1}),\\
\Delta_i^T &= \mathcal{I}(q_{i-1},\lambda p_i), & \tilde{\Delta}_i^T &= (2\mathcal{I}-\mathcal{G}_N)\,(\tilde{q}_{i-1},\lambda\tilde{p}_i),
\end{aligned}$$

where $\mathcal{I}$ and $\mathcal{G}_N$ are defined by (1.5.17) and (1.5.18), respectively. Hence, the recursions (1.4.21) and (1.5.28), together with (1.5.27) and Corollary 1.5.2, yield

$$\begin{aligned}
&\tilde{\Omega}_i=\Omega_i, & & & &1\le i\le N,\\
&\tilde{\Gamma}_i=\Gamma_i, & &\tilde{\Delta}_i=\Delta_i, & &1\le i\le N-1,\\
&\tilde{p}_i=p_i, & &\tilde{q}_i=q_i, & &0\le i\le N-1,
\end{aligned}$$

from which we conclude that

$$\begin{aligned}\tilde{p}_N\tilde{\Gamma}_N &= \lambda\tilde{p}_{N-1} - \tilde{p}_{N-1}\tilde{\Omega}_N - \tilde{p}_{N-2}\tilde{\Delta}_{N-1}^T \\ &= \lambda p_{N-1} - p_{N-1}\Omega_N - p_{N-2}\Delta_{N-1}^T = p_N\Gamma_N, \\ \tilde{q}_N\tilde{\Delta}_N &= \lambda\tilde{q}_{N-1} - \tilde{q}_{N-1}\tilde{\Omega}_N^T - \tilde{q}_{N-2}\tilde{\Gamma}_{N-1}^T \\ &= \lambda q_{N-1} - q_{N-1}\Omega_N^T - q_{N-2}\Gamma_{N-1}^T = q_N\Delta_N.\end{aligned} \tag{1.5.31}$$

Thus,

$$\begin{aligned}\tilde{\Gamma}_N &= (2\mathcal{I} - \mathcal{G}_N)(\tilde{q}_N, \lambda\tilde{p}_{N-1}) = 2\mathcal{I}(\tilde{q}_N, \lambda\tilde{p}_{N-1}) - \mathcal{G}_N(\tilde{q}_N, \lambda\tilde{p}_{N-1}) \\ &= 2\left(\Delta_N\tilde{\Delta}_N^{-1}\right)^T \mathcal{I}(q_N, \lambda p_{N-1}) = 2\left(\Delta_N\tilde{\Delta}_N^{-1}\right)^T \Gamma_N,\end{aligned}$$

where we have used that $\mathcal{G}_N(\tilde{q}_N, \lambda\tilde{p}_{N-1}) = \mathbf{0}$. This follows from the fact that

$$\tilde{q}_N^T(\theta_r^{(N)})\boldsymbol{u}_r^{(N)} = \left(\Delta_N\tilde{\Delta}_N^{-1}\right)^T q_N^T(\theta_r^{(N)})\boldsymbol{u}_r^{(N)} = \mathbf{0}, \qquad 1 \le r \le kN,$$

which is a consequence of Theorem 1.4.6. Therefore, $\tilde{\Delta}_N^T\tilde{\Gamma}_N = 2\Delta_N^T\Gamma_N$. There is some freedom in choosing the blocks $\tilde{\Gamma}_N$ and $\tilde{\Delta}_N$. We will choose

$$\tilde{\Gamma}_N = \sqrt{2}\Gamma_N, \quad \tilde{\Delta}_N = \sqrt{2}\Delta_N. \tag{1.5.32}$$

To show that $\tilde{\Omega}_{N+1} = \Omega_{N+1}$, we first observe that in view of (1.5.32) we have

$$\Gamma_N\tilde{\Gamma}_N^{-1} = \Delta_N\tilde{\Delta}_N^{-1} = (1/\sqrt{2})I_k,$$

which by (1.5.31) implies that $\tilde{p}_N = \left(1/\sqrt{2}\right)p_N$ and $\tilde{q}_N = \left(1/\sqrt{2}\right)q_N$. Therefore,

$$\tilde{\Omega}_{N+1} = (2\mathcal{I} - \mathcal{G}_N)(\tilde{q}_N, \lambda\tilde{p}_N) = \mathcal{I}(q_N, \lambda p_N) = \Omega_{N+1}.$$

Thus, similarly as in the symmetric case, the matrix $\tilde{J}_{N+1}$ given by (1.5.29)) can be obtained from the matrix J_{N+1} associated with the nonsymmetric $(N+1)$-block Gauss rule by multiplying the last off-diagonal blocks Γ_N and Δ_N by $\sqrt{2}$.

1.5.8 Bilinear forms

Given $A \in \mathbb{R}^{n\times n}$ we can define a bilinear form as a relation of the kind

$$\boldsymbol{u}^T f(A)\boldsymbol{v}, \qquad \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n. \tag{1.5.33}$$

Let us first suppose that A is symmetric and $\boldsymbol{v} = \boldsymbol{u}$, with $\|\mathbf{u}\| = 1$. Then, using the spectral decomposition (1.1.1), the expression can be written as a Stieltjes integral

$$\mathbf{u}^T f(A)\mathbf{u} = \sum_{i=1}^n f(\lambda_i)\omega_i^2 = \int f(t)d\omega(t), \tag{1.5.34}$$

where ω is a piece-wise constant distribution function with jumps at the eigenvalues λ_i of A. As shown in Subsection 1.5.1–1.5.2, a lower bound for the integral can be computed using a k-point Gauss quadrature rule (1.5.3) and an upper bound using a $(k+1)$-point Gauss–Radau quadrature rule (1.5.7).

If the Lanczos method breaks down, that is, if β_{k+1} in (1.4.2) vanishes, then the spectrum of T_k is a subset of the spectrum of A and the Gauss rule (1.5.4) yields the exact value of the bilinear form (1.5.33).

Pairs of Gauss and Gauss-Radau quadrature rules can be applied to compute bounds for expressions of the form (1.5.33) for any functions that are analytic on the convex hull of support of the measure and whose derivatives are of constant sign on this set. The situation when $\mathbf{u} \neq \mathbf{w}$ can be handled by writing (1.5.33) in the form

$$\mathbf{u}^T f(A)\mathbf{w} = \frac{1}{4}\left((\mathbf{u}+\mathbf{w})^T f(A)(\mathbf{u}+\mathbf{w}) - (\mathbf{u}-\mathbf{w})^T f(A)(\mathbf{u}-\mathbf{w})\right). \tag{1.5.35}$$

We now discuss how to compute upper and lower bounds for bilinear form of the kind

$$\boldsymbol{u}^T f(\sqrt{AA^T})\boldsymbol{v}, \qquad \text{or} \qquad \boldsymbol{u}^T f(\sqrt{A^T A})\boldsymbol{v}, \qquad\qquad \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n, \tag{1.5.36}$$

with the aid of Gauss quadrature rules. The approach is based on the partial Golub–Kahan bidiagonalization of the matrix A. A thorough discussion of this approach can be found in [24].

As above, the bilinear form (1.5.36) can be seen as a Stieltjes integral

$$\boldsymbol{u}^T f(\sqrt{AA^T})\boldsymbol{v} = \int f(\sqrt{t})d\omega(t), \tag{1.5.37}$$

where ω is a piecewise constant step function with jumps at the eigenvalues σ_j^2 of AA^T. The integral is over the support of the measure, i.e., over the interval $[\sigma_n^2, \sigma_1^2]$. The identity (1.5.37) is obtained by substituting the spectral factorization of AA^T into the left-hand side. It is natural to approximate (1.5.37) by using the small tridiagonal matrix (1.4.6). Golub and Meurant [58, 59] observed that $\boldsymbol{u}^T f(\sqrt{AA^T})\boldsymbol{v}$ is an ℓ-point Gauss quadrature rule for the approximation of (1.5.37); see also [24]. If all derivatives of the function $f(t)$ are positive for $t \geq 0$, then the remainder term for Gauss quadrature yields

$$\boldsymbol{u}^T f(\sqrt{AA^T})\boldsymbol{v} - \boldsymbol{u}^T f(\sqrt{T_\ell})\boldsymbol{v} = \frac{1}{(2\ell)!}\left(\frac{d^{2\ell}}{dt^{2\ell}} f(\sqrt{\theta})\right) \int \prod_{j=1}^{\ell} (t-\theta_j^{(\ell)})^2 d\omega(t), \tag{1.5.38}$$

where $\sigma_n^2 < \theta_1^{(\ell)} < \theta_2^{(\ell)} < \cdots < \theta_\ell^{(\ell)} < \sigma_1^2$ are the nodes of the quadrature rule and $\sigma_n^2 < \theta < \sigma_1^2$; see, e.g., Gautschi [53]. It follows that

$$\boldsymbol{u}^T f(\sqrt{AA^T})\boldsymbol{v} - \boldsymbol{u}^T f(\sqrt{T_\ell})\boldsymbol{v} > 0.$$

Consequently, the Gauss rule provides a lower bound for (1.5.37). Moreover, it is fairly easy to show that the lower bound is strictly increasing with ℓ; see [84] for details.

The remainder term for an $(\ell+1)$-point Gauss–Radau quadrature rule with ℓ "free" nodes and one fixed node at σ_1^2 is given by

$$\frac{1}{(2\ell+1)!}\left(\frac{d^{2\ell+1}}{dt^{2\ell+1}} f(\sqrt{\theta})\right) \int (t-\sigma_1^2) \prod_{j=1}^{\ell} (t-\hat{\theta}_j^{(\ell)})^2 d\omega(t),$$

where $\sigma_n^2 < \hat{\theta}_1^{(\ell)} < \hat{\theta}_2^{(\ell)} < \cdots < \hat{\theta}_\ell^{(\ell)} < \sigma_1^2$ denote the free nodes and $\sigma_n^2 < \theta < \sigma_1^2$; see [53]. It is clear that the remainder term is negative, i.e.,

$$\boldsymbol{u}^T f(\sqrt{AA^T})\boldsymbol{v} - \boldsymbol{u}^T f(\sqrt{\hat{T}_{\ell+1}})\boldsymbol{v} < 0.$$

It follows that the Gauss–Radau rule with a fixed node at σ_1^2 provides an upper bound for (1.5.37). It can be shown that the upper bound is strictly decreasing with ℓ; see [84].

The $(\ell+1)$-point Gauss–Radau quadrature rule with a fixed node at σ_1^2 can be expressed with a symmetric tridiagonal matrix $\hat{T}_{\ell+1} \in \mathbb{R}^{(\ell+1)\times(\ell+1)}$, whose elements, except for the last diagonal entry, are those of $B_{\ell+1,\ell}B_{\ell+1,\ell}^T$. The last diagonal entry of $\hat{T}_{\ell+1}$ is determined so that the matrix has the eigenvalue σ_1^2. This entry can be computed in only $\mathcal{O}(\ell)$ arithmetic floating point operations; see [58, 59] for details.

Let $\mathbf{w}_1$ and $\mathbf{w}_2$ be linearly independent vectors in $\mathbb{R}^n$. Then

$$\begin{aligned}\mathbf{w}_1^T f(\sqrt{AA^T})\mathbf{w}_2 = \frac{1}{4}(\mathbf{w}_1+\mathbf{w_2})^T f(\sqrt{AA^T})(\mathbf{w}_1+\mathbf{w}_2)\\ -\frac{1}{4}(\mathbf{w}_1-\mathbf{w}_2)^T f(\sqrt{AA^T})(\mathbf{w}_1-\mathbf{w}_2).\end{aligned} \tag{1.5.39}$$

In the rare event that the recursion formulas for Golub–Kahan bidiagonalization break down, the Gauss quadrature rule gives the exact value (in the absence of round-off errors).

1.5.9 Block methods

This section discusses the inexpensive computation of bounds or estimates of bounds for expressions of the form

$$W^T f(A)W, \tag{1.5.40}$$

where $W \in \mathbb{R}^{m\times k}$ has orthonormal columns with $1 \le k \ll m$. We are also interested in the computation of estimates of bounds for more general expressions

$$W^T f(A)V, \tag{1.5.41}$$

where the large matrix $A \in \mathbb{R}^{m\times m}$ may be nonsymmetric and $W, V \in \mathbb{R}^{m\times k}$ satisfy $V^T W = I_k$.

Golub and Meurant [58, 59] discuss how the application of a few steps of the symmetric block Lanczos method to a symmetric matrix A with initial block vector W yields an approximation of (1.5.40), and show that this approximation can be interpreted as a Gauss-type quadrature rule with respect to a discrete matrix-valued measure. In the special case when the block-size $k = 1$, the symmetric block Lanczos method simplifies to the standard symmetric Lanczos method. We showed in Subsection 1.5.1 that if $k = 1$ and A is symmetric and the function f has derivatives of constant sign in the convex hull of the spectrum of A, that pairs of suitable Gauss and Gauss–Radau rules yield upper and lower bounds for (1.5.40). Unfortunately, these quadrature rules are not guaranteed to yield upper and lower bounds when the pertinent derivatives of f change sign in the convex hull of the spectrum of A, and neither are block versions (with block-size $k > 1$) of the mentioned Gauss-type quadrature rules.

The matrix function (1.5.41) with a possibly nonsymmetric matrix A can be approximated by a function of a smaller matrix by application of a few steps of the nonsymmetric block Lanczos method with initial block vectors V and W. The reduction can be interpreted as a Gauss-type quadrature rule. However, generally this rule is not guaranteed to furnish upper or lower bounds for the elements of (1.5.41).

If the matrix $A \in \mathbb{R}^{m\times m}$ is symmetric, then the expression (1.5.40) can be written as a Stieltjes integral introducing the spectral factorization

$$A = Q\Lambda Q^T, \tag{1.5.42}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $\Lambda = \operatorname{diag}[\lambda_1, \ldots, \lambda_m]$. The eigenvalues are assumed to be ordered according to $\lambda_1 \leq \cdots \leq \lambda_m$. Substituting the spectral factorization (1.5.42) into (1.5.40) yields

$$W^T f(A) W = \widetilde{W} f(\Lambda) \widetilde{W}^T = \sum_{i=1}^{m} f(\lambda_i) \boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^T = \int f(\lambda) d\alpha(\lambda) =: \mathcal{I}f, \qquad (1.5.43)$$

where $\widetilde{W} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m] = W^T Q \in \mathbb{R}^{k\times m}$ and $\alpha : \mathbb{R} \to \mathbb{R}^{k\times k}$ is a discrete matrix-valued distribution with jumps $\boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^T$ at the eigenvalues λ_i of A.

We show that the quadrature rule $\mathcal{G}_N$ defined by (1.5.11) when $A = A^T$ is exact for all polynomials in $\mathbb{P}^{2N-1}$. Our proof is formulated entirely in terms of linear algebra and is shorter than existing proofs. The result was established in [58] and at about the same time extended in [119] to a more general (not necessarily discrete) class of matrix measures.

Theorem 1.5.5. *Let the function $\mathcal{I}$ be defined by (1.5.9) and the associated quadrature rule $\mathcal{G}_N$ by (1.5.11). Then $\mathcal{G}_N f = \mathcal{I}f$ for all $f \in \mathbb{P}^{2N-1}$.*

Proof. We first recall that the polynomials in the sets $\mathbb{P}^j$ have scalar coefficients. For fixed N, one can show by induction on the degree of p that

$$p(A)X_1 = X^{(N)} p(J_N) E_1, \quad p \in \mathbb{P}^{N-1},$$

where $X^{(N)} := [X_1, \ldots, X_N]$, where X_i are computed via the Symmetric Block Lanczos algorithm (1.4.8). Moreover,

$$\left(X^{(N)}\right)^T q(A) X_1 = q(J_N) E_1, \quad q \in \mathbb{P}^N.$$

Let $f \in \mathbb{P}^{2N-1}$. We may factor $f = pq$, where $p \in \mathbb{P}^{N-1}$ and $q \in \mathbb{P}^N$. Recalling that $X_1 = W$, we obtain

$$\begin{aligned} \mathcal{I}f &= W^T f(A) W = \left[X_1^T p(A)\right] q(A) X_1 \\ &= \left[E_1^T p(J_N) \left(X^{(N)}\right)^T\right] q(A) X_1 = E_1^T p(J_N) \left[\left(X^{(N)}\right)^T q(A) X_1\right] \\ &= E_1^T p(J_N) \left[q(J_N) E_1\right] = E_1^T f(J_N) E_1 = \mathcal{G}_N f. \end{aligned}$$

□

The following theorem is a generalization of Theorem 1.5.5. This result is also shown in [119].

Corollary 1.5.1. *Let $p(\lambda)$ and $q(\lambda)$ be polynomials with $k \times k$ matrix coefficients, and let $\mathcal{I}$ and $\mathcal{G}_N$ be defined by (1.4.11) and (1.5.13), respectively. Then $\mathcal{G}_N(p, q) = \mathcal{I}(p, q)$ when $\deg p + \deg q \leq 2N - 1$.*

Proof. The proof is related to the proof of Theorem 1.4.5. Let

$$p(\lambda) = \sum_{s=0}^{i} \lambda^s C_s, \qquad q(\lambda) = \sum_{t=0}^{j} \lambda^t D_t,$$

with $i+j \leq 2N-1$. By Theorem 1.5.5, we have

$$\begin{aligned}
\mathcal{G}_N(p,q) &= \mathcal{G}_N\left(\sum_{s=0}^{i}\lambda^s C_s, \sum_{t=0}^{j}\lambda^t D_t\right) = \sum_{s=0}^{i}\sum_{t=0}^{j} C_s^T \mathcal{G}_N\left(\lambda^s, \lambda^t\right) D_m \\
&= \sum_{s=0}^{i}\sum_{t=0}^{j} C_s^T \mathcal{G}_N\left(\lambda^{s+t}\right) D_t = \sum_{s=0}^{i}\sum_{t=0}^{j} C_s^T \mathcal{I}\left(\lambda^{s+t}\right) D_t \\
&= \sum_{s=0}^{i}\sum_{t=0}^{j} C_s^T \mathcal{I}\left(\lambda^s, \lambda^t\right) D_t = \sum_{s=0}^{i}\sum_{t=0}^{j} \mathcal{I}\left(\lambda^s C_s, \lambda^t D_t\right) \\
&= \mathcal{I}\left(\sum_{s=0}^{i}\lambda^s C_s, \sum_{t=0}^{j}\lambda^t D_t\right) = \mathcal{I}\left(p,q\right).
\end{aligned}$$

□

We establish results related to functions (1.5.41) that are analogous to those of the form (1.5.40).

Assume that $A \in \mathbb{R}^{m\times m}$ is diagonalizable and introduce the spectral factorization $A = Q\Lambda Q^{-1}$, where $Q \in \mathbb{C}^{m\times m}$ is nonsingular and $\Lambda = \mathrm{diag}\,[\lambda_1,\ldots,\lambda_m]$. When $A \neq A^T$, the eigenvalues λ_i may be complex-valued. Letting

$$\widetilde{W} = [\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_m] = W^T Q \in \mathbb{C}^{k\times m}, \quad \widetilde{V} = [\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_m] = \left(Q^{-1}V\right)^H \in \mathbb{C}^{k\times m},$$

we obtain

$$W^T f(A) V = \widetilde{W} f(\Lambda) \widetilde{V}^H = \sum_{i=1}^{m} f(\lambda_i)\boldsymbol{\alpha}_i\boldsymbol{\beta}_i^H =: \mathcal{I}f, \tag{1.5.44}$$

where the superscript ^{H} denotes transposition and complex conjugation.

Theorem 1.5.6. *Let the function $\mathcal{I}$ be defined by (1.4.20) and the associated quadrature rule $\mathcal{G}_N$ by (1.5.15). Then $\mathcal{G}_N f = \mathcal{I}f$ for all $f \in \mathbb{P}^{2N-1}$.*

Proof. The proof is similar to that of Theorem 1.5.5. Thus, we first show by induction on the degree of p that for fixed N,

$$\begin{aligned}
p(A^T)W_1 &= W^{(N)} p(J_N^T) E_1, \qquad & p \in \mathbb{P}^{N-1}, \\
\left(W^{(N)}\right)^T q(A) V_1 &= q(J_N) E_1, \qquad & q \in \mathbb{P}^N,
\end{aligned}$$

where $W^{(N)} := [W_1,\ldots,W_N]$.

Let $f \in \mathbb{P}^{2N-1}$. Factoring $f = pq$, where $p \in \mathbb{P}^{N-1}$ and $q \in \mathbb{P}^N$, yields similarly as in the proof of Theorem 1.5.5 that

$$\mathcal{I}f = W_1^T p(A) q(A) V_1 = E_1^T p(J_N) q(J_N) E_1 = E_1^T f(J_N) E_1 = \mathcal{G}_N f.$$

□

Corollary 1.5.2. *Let $p(\lambda)$ and $q(\lambda)$ be polynomials with $k\times k$ matrix coefficients, and let $\mathcal{I}(p,q)$ and $\mathcal{G}_N(p,q)$ be defined by (1.5.17) and (1.5.18), respectively. Then $\mathcal{G}_N(p,q) = \mathcal{I}(p,q)$ holds when $\deg p + \deg q \leq 2N-1$.*

Proof. The proof is similar to that of Corollary 1.5.1. □

We conclude this section with some comments on why pairs of N-block Gauss rules (1.5.11) and $(N+1)$-block anti-Gauss rules (1.5.24) may provide elementwise upper and lower bounds for (1.5.9) when the integrand is analytic in a sufficiently large region in the complex plane that contains the support of the measure $d\alpha$. Assume for notational simplicity that there are infinitely many orthogonal polynomials (1.5.10) and that f admits a representation of the form

$$f(x) = \sum_{i=0}^{\infty} C_i p_i(x),$$

where the coefficients C_i are $k \times k$ matrices. Assuming that $\mathcal{I}(f,f)$ is finite, one can show that the matrices C_i must converge to zero as i increases. Moreover, we have

$$\begin{aligned}
\mathcal{I}f &= C_0, \\
\mathcal{G}_N f &= C_0 + \sum_{i=2N}^{\infty} C_i \mathcal{G}_N p_i = C_0 + C_{2N}\mathcal{G}_N p_{2N} + C_{2N+1}\mathcal{G}_N p_{2N+1} + \sum_{i=2N+2}^{\infty} C_i \mathcal{G}_N p_i, \\
\mathcal{H}_{N+1} f &= C_0 + \sum_{i=2N}^{\infty} C_i \mathcal{H}_{N+1} p_i \\
&= C_0 + C_{2N}\mathcal{H}_{N+1} p_{2N} + C_{2N+1}\mathcal{H}_{N+1} p_{2N+1} + \sum_{i=2N+2}^{\infty} C_i \mathcal{H}_{N+1} p_i \\
&= C_0 - C_{2N}\mathcal{G}_N p_{2N} - C_{2N+1}\mathcal{G}_N p_{2N+1} + \sum_{i=2N+2}^{\infty} C_i \mathcal{H}_{N+1} p_i,
\end{aligned}$$

where we have used (1.5.20). Now, if the coefficient matrices C_i decay in norm sufficiently rapidly with increasing i, then the approximations

$$\begin{aligned}
\mathcal{G}_N f - \mathcal{I}f &\approx C_{2N}\mathcal{G}_N p_{2N} + C_{2N+1}\mathcal{G}_N p_{2N+1}, \\
\mathcal{H}_{N+1} f - \mathcal{I}f &\approx -C_{2N}\mathcal{G}_N p_{2N} - C_{2N+1}\mathcal{G}_N p_{2N+1}
\end{aligned}$$

suggest that the componentwise errors of the quadrature rules $\mathcal{G}_N f$ and $\mathcal{H}_{N+1} f$ are roughly equal in magnitude and of opposite sign. The norm of the matrices C_i decays quickly to zero when i increases if f is analytic in a large simply connected region in the complex plane that contains the support of the measure $d\alpha$ and has its boundary far away from the support. An argument similar can be made regarding the computation of bounds for (1.4.20) via (1.5.15) and (1.5.30). Thus, for integrands for which an expansion in terms of biorthogonal polynomials converges quickly, pairs of N-block Gauss and $(N+1)$-block anti-Gauss quadrature rules typically provide entrywise upper and lower bounds.

1.6 Inverse Problems

In an *inverse problem* one is interested in obtaining general information on the internal structure of a physical system starting from measured data. In this sense, an inverse problem is the inverse of a forward problem in which one measures an output knowing the input. The relation between the unknown parameters and the measured data is given by a mathematical model that describes the system. Some examples include tomography, image restoration, remote sensing, signal processing, and so on.

The first distinction that has to be done is between *linear* and *nonlinear* inverse problems, depending on the relation between the parameters we want to determine and the data. In the first case, we can represent the problem by the equation $\boldsymbol{d} = F(\boldsymbol{x})$, where F is a nonlinear operator. In the second case, F is a linear operator, i.e., a matrix, and the equation becomes $\boldsymbol{d} = G\boldsymbol{x}$. In order to deal with an inverse problem properly, we need to reduce the influence of errors in the observations. This is usually done using more measurements than unknown parameters in the model. This approach gives rise to an *overdetermined* system of equations, that is, in general, $F : \mathbb{R}^n \to \mathbb{R}^m$ or $G \in R^{m\times n}$, in the nonlinear and linear case, respectively, with $m > n$. When, on the contrary, there are more unknowns than measured data, the system is said to be *underdetermined*, that is $m < n$. Problems of this kind are called *ill-posed*. This concept, introduced by Jacques Hadamard [66], is the opposite of the notion of well-posedness and states that a problem is ill-posed if the solution does not exist or it is not unique or it is not a continuous function of the data. In this case, an approach consists in solving the *least squares problem*

$$\min_{\boldsymbol{x}} \|\mathbf{r}(\boldsymbol{x})\|_2^2 = \min_{\boldsymbol{x}} \|F(\boldsymbol{x}) - \boldsymbol{d}\|_2^2, \tag{1.6.1}$$

where $r(\boldsymbol{x})$ is the residual vector. The solutions, if any, are called *least squares solutions*. Among these, the one with minimal norm is called *normal solution*. The ways of solving least squares problems are different depending of the fact that they are linear or nonlinear.

1.6.1 Linear least squares problems

The classical approach to minimize the norm of the residual is to solve the *normal equations*, that is

$$\min_{\boldsymbol{x}} \|G\boldsymbol{x} - \boldsymbol{d}\|_2^2 \iff G^T G\boldsymbol{x} = G^T\boldsymbol{d}. \tag{1.6.2}$$

This can be seen computing the gradient of (1.6.2) and imposing that $\boldsymbol{x}$ is a critical point. The solution is then given by

$$\boldsymbol{x} = G^\dagger \boldsymbol{d},$$

where $G^\dagger$ is the Moore–Penrose pseudoinverse of G [16]; if $m \geq n$ and G has full rank, then $G^\dagger = (G^T G)^{-1} G^T$ and the system can be solved with the aid of the Cholesky or the QR factorization; see [16].

1.6.2 Nonlinear least squares problems

Methods for solving such problems are iterative and at each step we need to solve a related linear least square problem.

The vector $\boldsymbol{x}^*$ is a local minimizer of (1.6.1) if and only if it is a stationary point, i.e., if $\mathbf{f}'(\boldsymbol{x}^*) = 0$, where $\mathbf{f}'(\boldsymbol{x})$ is the gradient of the function f, whose jth component is

$$[\mathbf{f}'(\boldsymbol{\sigma})]_j = \frac{\partial f(\boldsymbol{x})}{\partial x_j} = \sum_{i=1}^m r_i(\boldsymbol{x}) \frac{\partial r_i(\boldsymbol{x})}{\partial x_j}, \qquad j = 1, \ldots, n; \tag{1.6.3}$$

see, e.g., [16] for a complete treatment.

If f is differentiable and smooth enough, then the Taylor approximation

$$\mathbf{f}'(\boldsymbol{x} + \mathrm{s}) = \mathbf{f}'(\boldsymbol{x}) + \mathbf{f}''(\boldsymbol{x})\mathrm{s} + O(\|\mathrm{s}\|^2) \simeq \mathbf{f}'(\boldsymbol{x}) + \mathbf{f}''(\boldsymbol{x})\mathrm{s}$$

is valid for $\|\mathbf{s}\|$ sufficiently small, where

$$[\mathbf{f}''(\boldsymbol{x})]_{jk} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_j \partial x_k} = \sum_{i=1}^{m} \left(\frac{\partial r_i(\boldsymbol{x})}{\partial x_j} \frac{\partial r_i(\boldsymbol{x})}{\partial x_k} + r_i(\boldsymbol{x}) \frac{\partial^2 r_i(\boldsymbol{x})}{\partial x_j \partial x_k} \right) \tag{1.6.4}$$

is the Hessian of the function f.

Newton's method chooses the iterative step $\mathbf{s}_\ell$ by imposing that $\boldsymbol{x}^*$ is a stationary point of Taylor approximation, i.e., by solving the linear system

$$\mathbf{f}''(\boldsymbol{x}_\ell)\mathbf{s}_\ell = -\mathbf{f}'(\boldsymbol{x}_\ell). \tag{1.6.5}$$

The next iterate is then computed as $\boldsymbol{x}_{\ell+1} = \boldsymbol{x}_\ell + \mathbf{s}_\ell$. The analytic expression of the Hessian $\mathbf{f}''(\boldsymbol{x})$ is not always available; whenever it is, its computation often implies a large computational cost. To overcome this problem, one possibility is to resort to the **Gauss–Newton method**, which is based on substituting (1.6.5) by the least squares minimization of a linear approximation of $\mathbf{r}(\boldsymbol{x} + \mathbf{s})$.

Let $\mathbf{r}$ be Fréchet differentiable and $\boldsymbol{x}_k$ denote the current approximation, then we can write

$$\mathbf{r}(\boldsymbol{x}_{k+1}) \simeq \mathbf{r}(\boldsymbol{x}_k) + J(\boldsymbol{x}_k)\mathbf{s}_k,$$

where $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \mathbf{s}_k$ and $J(\boldsymbol{x})$ is the Jacobian of $\mathbf{r}(\boldsymbol{x})$, defined by

$$[J(\boldsymbol{x})]_{ij} = \frac{\partial r_i(\boldsymbol{x})}{\partial x_j}, \qquad i = 1, \dots, m, \ j = 1, \dots, n. \tag{1.6.6}$$

At each step k, $\mathbf{s}_k$ is the solution of the linear least squares problem

$$\min_{\mathbf{s} \in \mathbb{R}^n} \|\mathbf{r}(\boldsymbol{x}_k) + J_k \mathbf{s}\|, \tag{1.6.7}$$

where $J_k = J(\boldsymbol{x}_k)$ or some approximation.

Problem (1.6.7) is equivalent to the normal equation

$$J_k^T J_k \mathbf{s} = -J_k^T \mathbf{r}(\boldsymbol{x}_k), \tag{1.6.8}$$

from which we obtain the following iterative method

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \mathbf{s}_k = \boldsymbol{x}_k - J_k^\dagger \mathbf{r}(\boldsymbol{x}_k). \tag{1.6.9}$$

Using this notation, the gradient (1.6.3) and the Hessian (1.6.4) of $f(\boldsymbol{x})$ can be written as

$$\begin{aligned} \mathbf{f}'(\boldsymbol{x}) &= J(\boldsymbol{x})^T \mathbf{r}(\boldsymbol{x}), \\ \mathbf{f}''(\boldsymbol{x}) &= J(\boldsymbol{x})^T J(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x}) H_i(\boldsymbol{x}), \end{aligned} \tag{1.6.10}$$

where

$$[H_i(\boldsymbol{x})]_{jk} = \frac{\partial^2 r_i(\boldsymbol{x})}{\partial x_j \partial x_k}$$

is the Hessian of the ith residual $r_i(\boldsymbol{x})$. Then, the Gauss–Newton method (1.6.9) can be seen as a special case of Newton's method, obtained by neglecting the term $\sum_{i=1}^m r_i(\boldsymbol{x}) H_i(\boldsymbol{x})$ from (1.6.10). This term is small if either each $r_i(\boldsymbol{\sigma})$ is mildly nonlinear at $\boldsymbol{x}_k$, or the residuals $r_i(\boldsymbol{x}_k)$, $i = 1, ..., m$, are small. When the residuals $r_i(\boldsymbol{x}_k)$ are small, or when the problem is consistent ($\mathbf{r}(\boldsymbol{x}^*) = 0$), the Gauss–Newton method is expected to behave similarly to Newton's method. In particular, the local convergence rate will be quadratic for both methods. If the above conditions are not satisfied, the Gauss–Newton method may not converge. A discussion on the local convergence of Gauss–Newton method can be found in [112].

To ensure convergence, the **damped Gauss–Newton method** replaces the approximation (1.6.9) by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \mathbf{s}_k, \tag{1.6.11}$$

where α_k is a step length to be determined. Two ways to choose α_k are:

- The Armijo–Goldstein principle [105], which selects α_k as the largest number in the sequence 2^{-i}, $i = 0, 1, \dots,$ for which the following inequality holds

  $$\|\mathbf{r}(\boldsymbol{x}_k)\|^2 - \|\mathbf{r}(\boldsymbol{x}_k + \alpha_k \mathbf{s}_k)\|^2 \geq \frac{1}{2}\alpha_k \|J_k \mathbf{s}_k\|^2.$$

  where

  $$-J_k \mathbf{s}_k = P_{J_k}\mathbf{r}(\boldsymbol{x}_k) = J_k J_k^{\dagger}\mathbf{r}(\boldsymbol{x}_k)$$

  is the orthogonal projection on the range of J_k.

- choose α_k to be the solution of the minimization problem

  $$\min_{\alpha} \|\mathbf{r}(\boldsymbol{x}_k + \alpha \mathbf{s}_k)\|^2.$$

1.6.3 Regularization Methods

The difficult in the approach described in the previous subsection is that the data are affected by *noise*. If the problem is ill-posed, then find a solution can be very difficult. In this scenario, one can think that only ill-posed problems are the ones hard to solve. Unfortunately, even if a problem is well-posed, it can be *ill-conditioned*, that is, a small perturbation on the data $\boldsymbol{d}$ causes a large perturbation on the solution $\boldsymbol{x}$. The measure of how the error propagates is given by the *condition number* of the mathematical model and it is an intrinsic property of the problem, that is, it does not depend on the way of solving it. If the problem is linear, the condition number is defined as the ratio between the largest and the smallest singular value of the matrix G. In this thesis we do not make use of the condition number of a nonlinear operator. For a general definition see [127, Part III].

Ill-conditioned problems belong to one of the following classes:

Rank-deficient problems: In this kind of problems there is a gap between the large and the small singular values. In this case, even if the columns of G are linearly independent from the mathematical point of view, the presence of errors causes that they are *almost* linearly independent. In this situation, instead of the rank of the matrix, the notion of *numerical rank* is more useful. This measures the number of columns of G that are practically linearly independent with respect to some error level.

Discrete ill-posed problems: These problems arise form the discretization of ill-posed problems and they are characterized by the fact that the singular values of the matrix G decay gradually to zero. Therefore, there is no gap in the singular values spectrum.

In both cases, the typical approach is to substitute the ill-conditioned problem with a well-conditioned one, which approximates the original problem, and solve it. This way to proceed is known as *regularization*. We briefly report the most famous regularization methods. For a deep discussion see [69].

Regularization methods can be divided in two classes: direct methods, which generally involve a decomposition of the matrix G, and iterative methods, which use the matrix G only via matrix-vector multiplications with G and G^T.

Tikhonov regularization: For discrete ill-posed problems its general formulation is

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \{\|G\boldsymbol{x}-\boldsymbol{d}\|^2 + \mu^2\|M\boldsymbol{x}\|^2\}, \tag{1.6.12}$$

where μ is the *regularization parameter* and M is the *regularization matrix*. The parameter μ controls the balance between the norm of regularization term $\|M\boldsymbol{x}\|^2$ and the residual norm. The matrix M is typically either the identity matrix I or a $t\times n$ discrete approximation of a derivative operator.

For example D_1 and D_2 defined as

$$\begin{aligned} D_1 &= \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(n-1)\times n} \\ D_2 &= \begin{bmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & & \\ & & 1 & -2 & 1 \end{bmatrix} \in \mathbb{R}^{(n-2)\times n} \end{aligned} \tag{1.6.13}$$

are discrete approximations to the first and second derivative operators.

If $\mathcal{N}(G)\cap\mathcal{N}(M)=\{0\}$, the the Tikhonov solution is unique and is given by

$$\boldsymbol{x}_T = (G^TG+\mu^2M^TM)^{-1}G^T\boldsymbol{d}.$$

The Tikhonov solution can be computed both via a direct method or an iterative one.

TSVD: This regularization technique is based on a low-rank approximation of the matrix G. The best rank ℓ approximation ($\ell \leq \operatorname{rank}(G) = p$) to G according to the Euclidean norm can be easily obtained by the SVD decomposition $G = U\Gamma V^T$; see Theorem 1.2.1. This procedure allows us to replace the ill-conditioned matrix G with a well-conditioned rank-deficient matrix G_ℓ. The corresponding solution $\hat{\boldsymbol{x}}$ is known as the truncated SVD (TSVD) solution [71] and it can be expressed as

$$\hat{\boldsymbol{x}}^{(\ell)} = G_\ell^\dagger \mathbf{r} = \sum_{i=1}^{\ell} \frac{\mathbf{u}_i^T\mathbf{r}}{\gamma_i}\mathbf{v}_i, \tag{1.6.14}$$

where $\ell = 1,\dots,p$ is the regularization parameter, γ_i are the singular values ($\gamma_1 \geq \cdots \geq \gamma_\ell$), the singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are the orthogonal columns of U and V, respectively, and $\mathbf{r}$ is the residual vector.

To introduce a regularization matrix $M\in\mathbb{R}^{t\times n}$ ($t\leq n$) we need the Generalized Singular Value Decomposition (GSVD) [107]. The GSVD of the matrix pair (G,M) is the factorization

$$G = U\Sigma_J Z^{-1}, \qquad M = V\Sigma_M Z^{-1}, \tag{1.6.15}$$

where Σ_J, Σ_M are diagonal matrices, U, V are orthogonal matrices, and Z is nonsingular.

The general form of the diagonal matrices Σ_J and Σ_M in (1.6.15), having the same size of J and M, is more complicated than we need, so we analyze two cases we will use in the third chapter. In the case $m \geq n = p$, the two diagonal matrices are given by

$$\Sigma_J = \begin{bmatrix} 0 & 0 \\ C & 0 \\ 0 & I_{n-t} \end{bmatrix}, \qquad \Sigma_M = \begin{bmatrix} S & 0 \end{bmatrix},$$

where I_{n-t} is the identity matrix of size $n-t$ and

$$C = \operatorname{diag}(c_1, \dots, c_t), \qquad S = \operatorname{diag}(s_1, \dots, s_t),$$

with $c_i^2 + s_i^2 = 1$. The diagonal elements are ordered such that the *generalized singular values* $\gamma_i = c_i/s_i$ are nondecreasing with $i = 1, \dots, t$, that is

$$0 \leq c_1 \leq \cdots \leq c_t \leq 1, \qquad 1 \geq s_1 \geq \cdots \geq s_t > 0.$$

When $p = m < n$, we have

$$\Sigma_J = \begin{bmatrix} 0 & C & 0 \\ 0 & 0 & I_{n-t} \end{bmatrix}, \qquad \Sigma_M = \begin{bmatrix} I_{n-2m} & 0 & 0 \\ 0 & S & 0 \end{bmatrix},$$

where C and S are diagonal matrices of size $m-n+t$. Requiring this number to be positive poses a constraint on the size of M.

The truncated GSVD (TGSVD) solution $\tilde{\boldsymbol{x}}_\ell$ is then defined as

$$\tilde{\boldsymbol{x}}_\ell^{(\ell)} = \sum_{i=\overline{p}-\ell+1}^{\overline{p}} \frac{\mathbf{u}_{2m-p+i}^T \mathbf{r}}{c_i} \mathbf{z}_{n-p+i} + \sum_{i=\overline{p}+1}^{p} (\mathbf{u}_{2m-p+i}^T \mathbf{r})\, \mathbf{z}_{n-p+i}, \tag{1.6.16}$$

where $\ell = 0, 1, \dots, \overline{p}$ is the regularization parameter, $\overline{p} = t$ if $m \geq n$, and $\overline{p} = m - n + t$ if $m < n$.

CG: This method has been originally designed for solve large sparse linear systems with symmetric positive definite coefficient matrix. Our interest lies in its application to normal equations (1.6.2) whose coefficient matrix $G^T G$ is symmetric positive semidefinite. It has been observed [67] that the CG algorithm has an intrinsic regularizing effect when applied to the normal equations. Among the different implementations of this algorithm, the most stable has been shown to be the CGLS due to Hestenes and Stiefel [76]. It is initialized with the starting vector $\boldsymbol{x}^{(0)}$, $\mathbf{r}^{(0)} = G\boldsymbol{x}^{(0)} - \boldsymbol{d}$ and $\mathbf{c}^{(0)} = G^T \mathbf{r}^{(0)}$ and proceeds as follows:

1. $\eta_k = \|G^T \mathbf{r}^{(k-1)}\|_2^2 / \|G\mathbf{c}^{(k-1)}\|_2^2$,
2. $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + \eta_k \mathbf{c}^{(k-1)}$,
3. $\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} - \eta_k G\mathbf{c}^{(k-1)}$,
4. $\xi_k = \|G^T \mathbf{r}^{(k)}\|_2^2 / \|G^T \mathbf{r}^{(k-1)}\|_2^2$,
5. $\mathbf{c}^{(k)} = G^T \mathbf{r}^{(k)} + \xi_k \mathbf{c}^{(k-1)}$.

Here $\boldsymbol{x}^{(k)}$ and $\mathbf{r}^{(k)}$ are the solution and the residual vector at step k, respectively.

LSQR: When the CGLS algorithm is implemented via the Golub-Kahan bidiagonalization algorithm described in Subsection 1.4.4, the resulting algorithm is called LSQR. After k steps of the Golub-Kahan algorithm with starting vector $q_1 = \boldsymbol{d}$, the solution $\boldsymbol{x}^{(k)} \in \mathbb{R}^n$ is defined as

$$\boldsymbol{x}^{(k)} = P_k \mathbf{y}^{(k)} = \beta_1 P_k B_{k+1,k}^{\dagger} \boldsymbol{e}_1,$$

that is, $\mathbf{y}^{(k)} \in \mathbb{R}^k$ is the solution of the least squares problem

$$\min \|B_{k+1,k}\mathbf{y}^{(k)} - \beta_1 \boldsymbol{e}_1\|_2,$$

where $B_{k+1,k}$ is computed via (1.4.4), $\beta_1 = \|\boldsymbol{d}\|$ and $\boldsymbol{e}_1$ is the first vector of the canonical base of size $k+1$.

1.6.4 Choice of the regularization parameter

In the previous subsection we saw different ways to regularize ill-conditioned problems. All these regularization methods require a good choice of the regularization parameter in order to work properly. There are different ways to choose the values for the parameters μ, ℓ or k of the previous Subsection and they can be divided in methods which assume the knowledge of the error $\|\boldsymbol{e}\|$ and those for which this information is not required.

Among those of the first kind, the *discrepancy principle* due to Morozov is the most famous one [102]. If the problem is consistent, that is, $G\boldsymbol{x} = \boldsymbol{d}$, the discrepancy principle chooses the reguralization parameter μ such that

$$\|G\boldsymbol{x}_\mu - \boldsymbol{d}\| = \kappa_e, \qquad \text{with} \quad \|\boldsymbol{e}\|_2 \leq \kappa_e,$$

where κ_e is an priori upper bound for the error. If the regularization parameter is discrete, then the smallest value for which $\|G\boldsymbol{x}_\ell - \boldsymbol{d}\| \leq \kappa_e$ can be chosen.

If an accurate bound for $\|\mathbf{e}\|$ is not available, the discrepancy principle cannot be applied. A large number of methods for determining a regularization parameter in such a case have been introduced for linear inverse problems [69]. They are known as *heuristic* because it is not possible to prove convergence results for them, in the strict sense of the definition of a regularization method; see [41, Chapter 4]. Among these we will briefly describe the *Generalized Cross Validation*(GCV) and the *L-curve criterion.*

GCV: It is based on statistical considerations [57, 130]. In particular, this method chooses the regularization parameter μ which minimizes the GCV function

$$\mathcal{G} = \frac{\|G\boldsymbol{x}_\mu - \boldsymbol{d}\|_2^2}{(\operatorname{trace}(I_m - G(\mu)))^2},$$

where $G(\mu)$, called the *influence matrix*, is such that $G(\mu)\boldsymbol{d} = G\boldsymbol{x}_\mu$.

L-curve: Let us consider the curve

$$\{\log \|G\boldsymbol{x}_\mu - \boldsymbol{d}\|, \log \|M\boldsymbol{x}_\mu\|\}. \tag{1.6.17}$$

This is called *L-curve* because exhibits a typical L-shape in many discrete ill-posed problems. The L-curve criterion seeks to determine the regularization parameter by detecting the index μ of the point of the curve closer to the corner of the “L”. This choice produces a solution for which both the norm and the residual are fairly small. There are several papers showing examples in which the L-curve method systematically fails; see, e.g., [68, 129]. Nevertheless, it has been shown by numerical experiments that it provides a good estimation of the optimal regularization parameter in many inverse problems of applicative interest [72, 114].

Various methods have been proposed to determine the corner of the L-curve. The L-corner method considers a sequence of pruned L-curves, obtained by removing an increasing number of points, and constructs a list of candidate “vertices” produced by two different selection algorithms. The corner is chosen from this list by a procedure which compares the norms and the residuals of the corresponding solutions [72]. It is currently implemented in [70].

Other ways to determine the corner are the restricted Regińska method [113, 114], the residual L-curve [114, 115], and the hybrid quasi-optimality criterion [102, 114]. A new approach, based on the comparison of regularized solutions computed by both TSVD and the Tikhonov method, has been recently proposed in [78].

2. Complex networks

2.1 Graphs and Complex networks

Definition 2.1.1. *A* **graph** *(or a* **network***) G is a pair (V, E), where V is a finite set and $E \subseteq V \times V$. The elements of the set V are called* **vertices** *or* **nodes** *and those of the set E are called* **edges** *or* **arcs***.*

A **complex** network is a graph that satisfied particular properties related to its topology. In this thesis we do not take in to account neither random graphs, in which the nodes are connected randomly, nor complete graphs, in which each node is connected to all the other nodes. Figure 2.1 shows two examples of this kind of graphs.

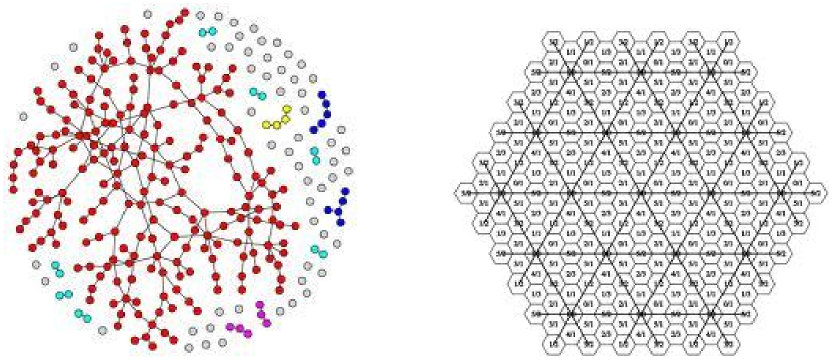


Figure 2.1: Two examples of graphs that are not complex networks. On the left side a random graph and on the right side a complete graph.

Two vertices are said to be **adjacent** if there is an edge connecting them. An edge is **incident** to the vertices that connects. The number of nodes adjacent to one node is the **degree** of that node. The set E is said to be symmetric if $(i, j) \in E \iff (j, i) \in E$. In this case the network is said to be **undirected**, **directed** otherwise. From this follows that in a directed network edges have an orientation. An out-edge at a node is an edge starting from that node and an in-edge is an edge that arrives at that node. A **walk** of length k is a sequence of vertices $v_1, v_2, \ldots, v_k$ such that there is an edge between vertex v_i and vertex v_{i+1} for $i = 1, 2, \ldots, k-1$. Vertices and edges may be repeated. An oriented walk is a walk in which every edge of the sequence is oriented from vertex v_i to vertex v_{i+1}. A closed walk is a walk in which the last node of the sequence coincides with the first node. A **path** is a walk with all vertices distinct. Finally, a **loop** is a closed walk of length one. The notion of path is at the base of the concept of connectness.

Definition 2.1.2. *An undirected network is* **connected** *if there is a path connecting any two nodes. A directed network is* **strongly connected** *if there is an oriented path that connects any pair of nodes,* **weakly connected** *if the edges in the path do not follow the same orientation.*

In a directed graph, an alternating walk of length k starting from an out-edge at node v_1 and ending at node v_{k+1} is a list of $k+1$ nodes such that there exists an edge from v_i to v_{i+1} if i is odd and an edge from v_{i+1} to v_i if i is even. Analogously, an alternating walk of length k, starting with an in-edge at node v_1 and ending at node v_{k+1} is a list of $k+1$ nodes such that there exists an edge from v_{i+1} to v_i if i is odd and an edge from v_i to v_{i+1} if i is even

A graph is said to be **simple** if it is undirected and does not contain loops and multiple edges and is **weighted** if each edge (i, j) has a weight $w_{i,j}$. Even if weighted networks are useful and common in real world, we do not consider them further in this thesis.

Definition 2.1.3 (Adjacency matrix)**.** *The adjacency matrix A of a unweighted simple graph G is defined as*

$$[A]_{i,j} = \begin{cases} 1 & \textit{if there is an edge from node } i \textit{ to node } j \\ 0 & \textit{otherwise} \end{cases} \tag{2.1.1}$$

The adjacency matrix is symmetric if and only if G is undirected.

From the definition of adjacency matrix it is easy to see that for $\ell \geq 1$, the (i, j) entry of A^ℓ gives the number of walks of length ℓ starting at node i and ending at node j.

As we will see in the next Section, the adjacency matrix can be used to investigate properties of the network. In real world there are different kind of interactions between entities that can be modelized and analyzed with the aid of the complex networks theory. In the past decades, many different models of networks have been introduced to this aim. We will briefly describe the most important ones.

Strongly clustered network One characteristic of real world network is the presence of many triangles, that is, closed walks of length three. This behavior is evident in social networks taking for example in to account the friendship as a relation between nodes. In this case, the probability that a friend of my friend is also my friend is high, giving rise to a triangle in the network. The *average clustering coefficient* is an indicator of the presence of such triangles within the network and is defined as the mean, computed over all the nodes, of the clustering coefficients of each node, that is:

$$C = \frac{1}{n}\sum_{i=1}^{n} C_i \qquad \text{where} \qquad C_i = \frac{\#\text{ of triangles based on node } i}{\#\text{ of possible triangles based on node } i}.$$

Scale-free network Generally, if we consider a network with n nodes generated at random connecting two nodes with probability p, then the probability $P(k)$ that a particular node has degree k follows a binomial distribution

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}.$$

However, as we pointed out at the beginning of this Section, random networks are not interesting when we want to model real world interactions. A more useful model is the *scale-free* network [9] in which the probability $P(k)$ follows a power-law distribution $P(k) \sim k^{-\gamma}$, where typically $2 < \gamma < 3$. This special behavior of degree distribution appears when a new node in a network is preferentially attached to nodes with high degree. For this reason, this kind of networks are also known as *preferential attachment* or the *rich-get-richer* models.

Small world network The small world network is a graph in which the distance L, that is, the shortest path, between two nodes chosen at random grows proportionally to the logarithm of the number of nodes n, i.e., $L \propto \log n$. The characteristic of this kind of networks is that every node can be visited with a small number of steps. A model has been introduced by Watts and Strogatz [133]: starting from a regular network we choose a node at random and rewire an edge with probability p. This allows us to obtain a network in which both the clustering coefficient C is higher than in the random model and the distance L is lower than in the regular model. The rewiring process is shown in Figure 2.2.

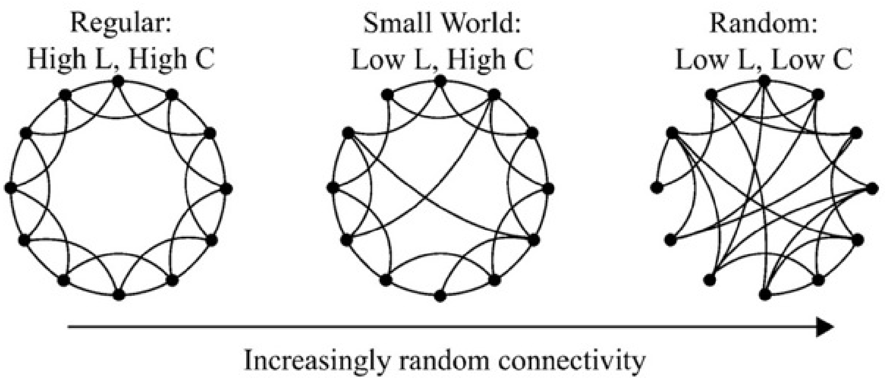


Figure 2.2: Representation of the Watts-Strogatz rewiring process.

2.2 Centrality indices and rank of the nodes

Given a large graph, it can be useful to extract numerical quantities that describe interesting global properties of the graph, such as the overall importance of a particular node within the network, or the ease of traveling from one node to another. In an undirected network, the *degree* of a node i, which is the number of nodes connected to that node, provides a rough measure of the importance of the node. However, this quantity fails to take into consideration the importance of the nodes connected to node i, e.g., how well-connected they are. Analogously, a rough measure of the ease of traveling from node i to node j is the length of the *shortest path* connecting these nodes. However, this measure fails to take into consideration the possibility that a somewhat longer path may be useful. For example, during rush hour many commuters resort to longer routes to reduce their exposure to traffic. Similarly, the expected time required by a virus to travel from Alice to Bob decreases as the number of their mutual friends increases, and decreases further as the number of friendships between mutual friends increases. Even if Alice and Bob are friends, i.e., a path of length one exists between them, the virus may be transmitted via one or more mutual friends along a path of length larger than one.

The following observation establishes an important link between graph theory and linear algebra, a connection that has been exploited by Estrada and his collaborators in their quest for alternatives to the concepts of degree and shortest path; see, e.g., [44–49]. Given a function

$$f(A) = \sum_{\ell=0}^{\infty} c_\ell A^\ell \tag{2.2.1}$$

with nonnegative coefficients c_ℓ chosen to guarantee convergence, the quantity $[f(A)]_{ij}$ can be interpreted as a measure of the overall ease of traveling from node i to node j within the network. The term $c_0 I_m$ has no specific meaning and is introduced for convenience. We may think of the coefficients c_ℓ as weights. They are chosen to decrease as ℓ increases in order to penalize the contributions of long walks and to secure convergence of the sum (2.2.1). The choice $c_\ell = 1/\ell!$ yields

$$f(A) = \exp(A) \tag{2.2.2}$$

and is discussed by Estrada and Higham [46]. Another popular choice are coefficients that yield functions of the form

$$f(A) = (I - cA)^{-1}, \tag{2.2.3}$$

where c is a coefficient small enough so that the representation (2.2.1) exists; see [19, 46].

Estrada and his collaborators have defined the following quantities, relevant to both directed and undirected graphs:

- The *f-communicability* [46] from node j to node i, given by $[f(A)]_{ij}$, quantifies the ease of traveling from node i to node j.

- The *f-communicability betweenness* [47] of node r is given by

$$\frac{1}{(m-1)(m-2)} \sum_{i \neq r} \sum_{\substack{j \neq r \\ j \neq i}} \frac{[f(A)]_{ij} - [f(A_r)]_{ij}}{[f(A)]_{ij}}, \tag{2.2.4}$$

  where A_r is the adjacency matrix of the graph obtained by removing from G all edges involving node r. This is a measure of the amount of communication passing through node r.

- The *average f-communicability* from node r is defined by

$$\frac{1}{m-1} \boldsymbol{e}_r^T f(A) \boldsymbol{c}_r,$$

  where $\boldsymbol{e}_r = [0, \dots, 0, 1, 0, \dots, 0]^T$ is the rth axis vector, $\boldsymbol{c} = [1, 1, \dots, 1]^T$ is the vector with all entries equal to one, and $\boldsymbol{c}_r = \boldsymbol{c} - \boldsymbol{e}_r$. This quantity is defined in [45] for $f(A) = \exp(A)$.

The above quantities are applied to symmetric matrices in [45–47], but they are of interest for nonsymmetric adjacency matrices, which correspond to directed graphs, as well. In the case of a directed or undirected graph, the f-communicability from node i to itself, i.e.,

$$[f(A)]_{ii} = \boldsymbol{e}_i^T f(A)\boldsymbol{e}_i, \tag{2.2.5}$$

is referred to as the *f-subgraph centrality* of node i; see [46, 48, 49]. Further quantities relevant for undirected graphs are discussed in [11]. The following quantities also would appear to be of interest:

- The *f-starting convenience* of node i, given by

  $$m\frac{\boldsymbol{e}_i^T f(A)\boldsymbol{c}}{\boldsymbol{c}^T f(A)\boldsymbol{c}},$$

  quantifies the ease of traveling from node i to anywhere in the network. This is the sum of the communicabilities from node i to all other nodes, scaled so that the average of the quantity over all nodes is one.

- The *f-ending convenience* of node i, given by

  $$m\frac{\boldsymbol{c}^T f(A)\boldsymbol{e}_i}{\boldsymbol{c}^T f(A)\boldsymbol{c}},$$

  quantifies the ease of traveling to node i from anywhere in the network. This is the sum of the communicabilities from all other nodes to node i, scaled so that the average of the quantity over all nodes is one. The f-ending convenience agrees with the f-starting convenience when the graph G is undirected.

- The *alternative f-communicability betweenness* of node r is given by

  $$\frac{\boldsymbol{c}_r^T f(A)\boldsymbol{c}_r - \boldsymbol{c}_r^T f(A_r)\boldsymbol{c}_r}{\boldsymbol{c}_r^T f(A)\boldsymbol{c}_r}. \tag{2.2.6}$$

  This quantity is related to (2.2.4), but differs from the latter in that it takes into consideration the effect of removing node r on the diagonal elements of $f(A)$, i.e., it takes into account the importance of node r as an intermediate step in closed walks. The scaling is also different from (2.2.4), as the summation for the latter quantity includes the *relative* change in each value $[f(A)]_{ij}$ (with $i \neq r$, $j \neq r$, $i \neq j$) caused by the removal of all edges involving node r, whereas all corresponding terms in (2.2.6) are divided by the same number. A reason for introducing (2.2.6) is that, different from (2.2.4), it can be conveniently approximated by (block) Gauss-type quadrature rules. By construction, both quantities (2.2.4) and (2.2.6) are between 0 and 1.

We generally suppress the prefix "f-" in the above quantities when the function f is clear from the context.

2.3 Undirected networks

Let G be an undirected and unweighted graph without loops or multiple edges. We assume that n, the number of nodes of G, is large and that the number of edges is much smaller than n^2. Networks that can be represented by this kind of graph arise in many scientific and industrial applications, including genetics, epidemiology, energy distribution, and telecommunications; see [13, 43, 46, 103].

Note that the quantities defined in Section 2.2 are of the form

$$\mathbf{u}^T f(A)\mathbf{w} \tag{2.3.1}$$

with $\mathbf{u}$ and $\mathbf{w}$ different or the same vector. In many applications, $\mathbf{u}$ is the jth axis vector $\mathbf{e}_j = [0, \dots, 0, 1, 0, \dots, 0]^T \in \mathbb{R}^n$.

When the adjacency matrix A is large, i.e., when the graph G has many nodes, direct evaluation of $f(A)$ generally is not feasible. Benzi and Boito [10] applied pairs of Gauss and Gauss-Radau rules to compute upper and lower bounds for selected entries of $f(A)$. This work is based on the connection between the symmetric Lanczos process, orthogonal polynomials, and Gauss-type quadrature, explored by Golub and his collaborators in many publications; see Golub and Meurant [59] for details and references. A brief review of this technique is provided in Section 1.5. An application of pairs of block Gauss-type quadrature rules to simultaneously determine approximate upper and lower bounds for several entries of $f(A)$ is described in [50].

The main drawback of quadrature-based methods is that the computational effort is proportional to the number of desired bounds. Quadrature-based methods are attractive to use when bounds for only a few quantities (2.3.1) are to be computed. However, these methods can be expensive to use when bounds for many expressions are to be evaluated. This situation arises, for instance, when we would like to determine one or a few nodes with the largest f-subgraph centrality in a large graph, because this requires the computation of upper and lower bounds for all diagonal entries of $f(A)$.

In the next Section, we describe how to bound quantities of the form (2.3.1) by using a low-rank approximation of the adjacency matrix A.

2.3.1 Use of low-rank approximation

We derive bounds for expressions of the kind $\mathbf{u}^T f(A)\mathbf{w}$, with $\|\mathbf{u}\| = \|\mathbf{w}\| = 1$, in terms of a partial spectral factorization of A. The function f is assumed to be nondecreasing and nonnegative on the spectrum of A. This is true, for example, for the functions (2.2.2) and (2.2.3).

Introduce the spectral factorization

$$A = V\Lambda V^T,$$

where the eigenvector matrix $V = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n] \in \mathbb{R}^{n\times n}$ is orthogonal and the eigenvalues in the diagonal matrix $\Lambda = \operatorname{diag}[\lambda_1, \lambda_2, \dots, \lambda_n] \in \mathbb{R}^{n\times n}$, are ordered according to $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. By definition,

$$f(A) = Vf(\Lambda)V^T = \sum_{k=1}^{n} f(\lambda_k)\mathbf{v}_k\mathbf{v}_k^T, \tag{2.3.2}$$

so that

$$f_{\mathbf{u},\mathbf{w}}(A) := \mathbf{u}^T f(A)\mathbf{w} = \sum_{k=1}^{n} f(\lambda_k)\tilde{u}_k\tilde{w}_k,$$

where $\tilde{u}_k = \mathbf{u}^T\mathbf{v}_k$ and $\tilde{w}_k = \mathbf{w}^T\mathbf{v}_k$.

Let the first N eigenpairs $\{\lambda_k, \mathbf{v}_k\}_{k=1}^N$ of A be known. Then $f_{\mathbf{u},\mathbf{w}}(A)$ can be approximated by

$$\mathbf{u}^T f(A)\mathbf{w} \approx F_{\mathbf{u},\mathbf{w}}^{(N)} := \sum_{k=1}^{N} f(\lambda_k)\tilde{u}_k\tilde{w}_k. \tag{2.3.3}$$

The following result shows how upper and lower bounds for $f_{\mathbf{u},\mathbf{w}}(A)$ can be determined with the aid of the first N eigenpairs of A.

Theorem 2.3.1. *Let the function f be nondecreasing and nonnegative on the spectrum of A and let $F_{\mathbf{u},\mathbf{w}}^{(N)}$ be defined by (2.3.3). Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ be the N largest eigenvalues of A and let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ be associated orthonormal eigenvectors. Then we have*

$$L_{\mathbf{u},\mathbf{w}}^{(N)} \leq f_{\mathbf{u},\mathbf{w}}(A) \leq U_{\mathbf{u},\mathbf{w}}^{(N)}, \tag{2.3.4}$$

where

$$L_{\mathbf{u},\mathbf{w}}^{(N)} := F_{\mathbf{u},\mathbf{w}}^{(N)} - f(\lambda_N)\left(1 - \sum_{k=1}^{N}\tilde{u}_k^2\right)^{1/2}\left(1 - \sum_{k=1}^{N}\tilde{w}_k^2\right)^{1/2},$$

$$U_{\mathbf{u},\mathbf{w}}^{(N)} := F_{\mathbf{u},\mathbf{w}}^{(N)} + f(\lambda_N)\left(1 - \sum_{k=1}^{N}\tilde{u}_k^2\right)^{1/2}\left(1 - \sum_{k=1}^{N}\tilde{w}_k^2\right)^{1/2}.$$

Proof. The Cauchy inequality yields

$$\begin{aligned}\left|f_{\mathbf{u},\mathbf{w}}(A) - F_{\mathbf{u},\mathbf{w}}^{(N)}\right| &= \left|\sum_{k=N+1}^{n} f(\lambda_k)\tilde{u}_k\tilde{w}_k\right| \leq f(\lambda_N)\sum_{k=N+1}^{n}|\tilde{u}_k|\,|\tilde{w}_k| \\ &\leq f(\lambda_N)\left(\sum_{k=N+1}^{n}\tilde{u}_k^2\right)^{1/2}\left(\sum_{k=N+1}^{n}\tilde{w}_k^2\right)^{1/2} \\ &= f(\lambda_N)\left(1 - \sum_{k=1}^{N}\tilde{u}_k^2\right)^{1/2}\left(1 - \sum_{k=1}^{N}\tilde{w}_k^2\right)^{1/2},\end{aligned}$$

from which (2.3.4) follows. □

Corollary 2.3.1. *Assume that the conditions of Theorem 2.3.1 hold and let $\mathbf{u} = \mathbf{w}$. Then*

$$F_{\mathbf{u},\mathbf{u}}^{(N)} \leq f_{\mathbf{u},\mathbf{u}}(A) \leq U_{\mathbf{u},\mathbf{u}}^{(N)}. \tag{2.3.5}$$

Proof. We have

$$0 \leq f_{\mathbf{u},\mathbf{u}}(A) - F_{\mathbf{u},\mathbf{u}}^{(N)} = \sum_{k=N+1}^{n} f(\lambda_k)\tilde{u}_k^2 \leq f(\lambda_N)\sum_{k=N+1}^{n}\tilde{u}_k^2 = f(\lambda_N)\left(1 - \sum_{k=1}^{N}\tilde{u}_k^2\right),$$

which implies (2.3.5). □

Corollary 2.3.2. *Let the conditions of Theorem 2.3.1 hold. Then the bounds (2.3.4) satisfy,*

$$\left|f_{\mathbf{u},\mathbf{w}}(A) - F_{\mathbf{u},\mathbf{w}}^{(N+1)}\right| \leq \left|f_{\mathbf{u},\mathbf{w}}(A) - F_{\mathbf{u},\mathbf{w}}^{(N)}\right| \tag{2.3.6}$$

and

$$\begin{array}{rcl} L^{(N)}_{\mathbf{u},\mathbf{w}} - F^{(N)}_{\mathbf{u},\mathbf{w}} & \leq & L^{(N+1)}_{\mathbf{u},\mathbf{w}} - F^{(N+1)}_{\mathbf{u},\mathbf{w}} \leq 0, \\ U^{(N)}_{\mathbf{u},\mathbf{w}} - F^{(N)}_{\mathbf{u},\mathbf{w}} & \geq & U^{(N+1)}_{\mathbf{u},\mathbf{w}} - F^{(N+1)}_{\mathbf{u},\mathbf{w}} \geq 0, \end{array} \tag{2.3.7}$$

for $1 \leq N < n$. *For the bounds (2.3.5), we have*

$$F^{(N)}_{\mathbf{u},\mathbf{u}} \leq F^{(N+1)}_{\mathbf{u},\mathbf{u}}, \qquad U^{(N)}_{\mathbf{u},\mathbf{u}} \geq U^{(N+1)}_{\mathbf{u},\mathbf{u}}, \qquad 1 \leq N < n. \tag{2.3.8}$$

Proof. The monotonic behavior of $N \to F^{(N)}_{\mathbf{u},\mathbf{u}}$ is a consequence of the nonnegativity of f on the spectrum of A, and the monotonic behavior of $N \to U^{(N)}_{\mathbf{u},\mathbf{u}}$ follows from the fact that f is a nondecreasing function. The inequalities (2.3.6) and (2.3.7) can be shown similarly. □

In the following we will show how to determine a set of nodes with the largest f-subgraph centrality $[f(A)]_{ii} = \mathbf{e}_i^T f(A)\mathbf{e}_i$ of a large network. The same arguments can be applied to find the most important nodes with respect other centrality measures. The inequalities (2.3.5) and (2.3.8) are important in this context. For notational convenience, we refer to the lower and upper bounds $F^{(N)}_{\mathbf{u},\mathbf{u}}$ and $U^{(N)}_{\mathbf{u},\mathbf{u}}$ in (2.3.5) as $L^{(N)}_{ii}$ and $U^{(N)}_{ii}$ when $\mathbf{u} = \mathbf{e}_i$.

The bounds (2.3.4) and (2.3.6) are relevant when we seek to determine pairs of nodes with the largest f-communicability. Letting $\mathbf{u} = \mathbf{e}_i$, $\mathbf{w} = \mathbf{c}/\|\mathbf{c}\| = n^{-1/2}[1, 1, \ldots, 1]^T$, and multiplying all the bounds by $\|\mathbf{c}\| = \sqrt{n}$, we can apply them to the f-starting convenience.

Determining important nodes by partial spectral factorization This section describes how knowledge of the N leading eigenpairs $\{\lambda_k, \mathbf{v}_k\}_{k=1}^N$ of A and the bounds (2.3.5) with $\mathbf{u} = \mathbf{e}_i$ can be used to determine a subset of nodes that contains the nodes with the largest f-subgraph centrality $[f(A)]_{ii}$. We refer to the nodes with the largest f-subgraph centrality as the most important nodes. The function f is required to be nondecreasing and nonnegative. We will comment on special computational issues that arise when f is the exponential function.

Let $L^{(N)}_{ii}$ and $U^{(N)}_{ii}$ be the lower and upper bounds defined after Corollary 2.3.2, and let $\mathcal{L}^{(N)}_m$ denote the mth largest lower bound $L^{(N)}_{ii}$. Introduce the index sets

$$S^{(N)}_m = \left\{ i \; : \; U^{(N)}_{ii} \geq \mathcal{L}^{(N)}_m \right\}, \qquad N = 1, 2, \ldots, n.$$

It is not hard to see that if $i \notin S^{(N)}_m$, then node i cannot be in the subset of the m nodes with the largest f-subgraph centrality. Moreover, any node whose index is an element of $S^{(N)}_m$ can be in this subset.

Let $|S^{(N)}_m|$ denote the cardinality of $S^{(N)}_m$. The following relations are useful in the sequel.

Corollary 2.3.3.

$$S^{(n)}_m \subseteq S^{(n-1)}_m \subseteq \cdots \subseteq S^{(1)}_m \qquad \textit{and} \qquad |S^{(n)}_m| \geq m. \tag{2.3.9}$$

The set $S^{(N)}_m$ *contains the indices for a subset of nodes that contains the set of the* m *most important nodes. In particular, when* $|S^{(N)}_m| = m$, *the set* $S^{(N)}_m$ *contains the indices for the* m *most important nodes.*

Proof. By the definition of the sets $S_m^{(N)}$, each set contains at least m indices. The relations (2.3.9) now follow from (2.3.8) with $\mathbf{u} = \mathbf{e}_i$ for $1 \leq i \leq n$. The observation about the situation when $|S_m^{(N)}| = m$ is a consequence of the fact that the set $S_m^{(N)}$ contains the indices for the m nodes with the largest f-subgraph centrality. □

Remark 2.3.1. *The lower bound $L_{ii}^{(N)}$ defined after Corollary 2.3.2 usually converges to $[f(A)]_{ii}$ much faster than the corresponding upper bound $U_{ii}^{(N)}$ as N increases. Therefore, for N fixed, $L_{ii}^{(N)}$ typically is a better approximation of $[f(A)]_{ii}$ than $\frac{1}{2}(L_{ii}^{(N)} + U_{ii}^{(N)})$. This is due to the fact that the lower bound is a truncation of the expansion (2.3.2), cf. (2.3.3) and (2.3.5), while the upper bound is obtained from the lower bound by adding a sufficiently large and computable quantity.*

Remark 2.3.2. *A particular ordering of the lower bounds $L_{ii}^{(N)}$, $i \in S_m^{(N)}$, does not have to correspond to the same ordering of the f-subgraph centralities $[f(A)]_{ii}$, i.e., an inequality $L_{ii}^{(N)} > L_{jj}^{(N)}$ for $j \in S_m^{(N)} \backslash \{i\}$ does not imply that the ith node has the largest f-subgraph centrality.*

We turn to some computational issues. Evaluation of the bounds (2.3.4) and (2.3.5) requires the computation of $f(\lambda_N)$. This may result in overflow when the graph contains many nodes and f is the exponential function (2.2.2). For instance, when the computations are carried out in "double precision arithmetic", i.e., with about 16 significant decimal digits, we obtain overflow when evaluating $\exp(x)$ for $x \gtrsim 710$. This difficulty can be circumvented by replacing A by $A - \mu I$, where I is the identity matrix and μ is an estimate of the largest eigenvalue of A. We then seek to approximate $[f(A - \mu I)]_{ii}$ instead of $[f(A)]_{ii}$, where we note that

$$[f(A)]_{ii} = \exp(\mu)\, [f(A - \mu I)]_{ii}.$$

Since $A \in \mathbb{R}^{n \times n}$ is an adjacency matrix for an unweighted undirected graph without loops, its spectral radius is bounded by $n - 1$. We therefore may use $\mu = n - 1$. However, since we determine a partial spectral factorization $\{\lambda_k, \mathbf{v}_k\}_{k=1}^N$ of A, the largest eigenvalue λ_1 of A is available. Therefore, we let $\mu = \lambda_1$ in the computed examples reported in Section 2.3.2.

Another computational difficulty to overcome is that we do not know in advance how the dimension N of the leading invariant subspace $\{\lambda_k, \mathbf{v}_k\}_{k=1}^N$ of A should be chosen in order to obtain useful bounds (2.3.4) or (2.3.5). We will use the restarted block Lanczos method `irbleigs` described in [3, 4] to compute invariant subspaces in the examples of Section 2.3.2. This method computes the leading invariant subspace $\{\lambda_k, \mathbf{v}_k\}_{k=1}^q$ of A of user-chosen dimension q. The method applies Leja shifts to damp unwanted eigenvector components in the block Krylov subspaces generated. These shifts are constructed in the same manner as Leja points, which are suitable interpolation points when approximating analytic functions by an interpolating polynomial in a region in the complex plane. If the computed bounds (2.3.4) or (2.3.5) are not tight enough, e.g., if $|S_m^{(q)}|$ is larger than m, then we restart the computations with `irbleigs` to determine the next q eigenpairs $\{\lambda_k, \mathbf{v}_k\}_{k=q+1}^{2q}$ of A to obtain the leading invariant subspace $\text{span}\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{2q}\}$ of A. This can be done by passing the already available q eigenvectors $\{\mathbf{v}_k\}_{k=1}^q$ of A to `irbleigs` using the option `opts.eigvec`. If $|S_m^{(N)}| = m$, for $N \leq 2q$, then we are done, otherwise we compute the next q eigenpairs $\{\lambda_k, \mathbf{v}_k\}_{k=2q+1}^{3q}$ of A with `irbleigs`, and so on. The approach outlined requires that all already computed eigenvectors be stored when determining the next batch of q

eigenvectors. This method for determining an invariant subspace of desired dimension is attractive when the computer used allows storage of an orthonormal basis for the already computed subspace. A comparison of `irbleigs` and the MATLAB function `eigs`, which is based on ARPACK [88], is reported in [4] and shows the former method to be competitive. The use of `irbleigs` is particularly advantageous when only few auxiliary vectors can be stored. This often is the case for large-scale problems. Moreover, `irbleigs` is a block method, while `eigs` implements a standard restarted Lanczos method. Block methods may perform more efficiently on many computers; see, e.g., Gallivan et al. [52].

When the adjacency matrix is very large and the dimension N of a leading invariant subspace $\{\lambda_k, \mathbf{v}_k\}_{k=1}^{N}$ such that $|S_m^{(N)}| = m$ is fairly large, the storage requirement for a basis for this subspace may be problematic. We describe an approach to handle this situation. Note that the evaluation of the bounds (2.3.4) and (2.3.5) does not require simultaneous access to all computed eigenvectors. We therefore may reduce the storage requirement by limiting the number of eigenvectors passed to `irbleigs`. This can be achieved by calling `irbleigs` with the option `opts.sigma` as follows. This option determines eigenvalues of A close to a specified value. We choose this value to be the smallest eigenvalue of the invariant subspace already computed. The option `opts.eigvec` is used to pass available eigenvectors associated with eigenvalues closest to the smallest computed eigenvalue. The number of eigenvectors passed, $M_{\max}$, is close to the largest possible number of eigenvectors that fit into fast computer memory. For definiteness, assume that q new eigenpairs with eigenvalues smaller than the smallest of the already computed eigenvalues are desired. Then this approach typically gives some new eigenpairs, which are used to update the bounds (2.3.4) and (2.3.5). Also a few previously already computed eigenpairs may be recomputed. The outlined computations only require storage of $M_{\max} + q$ eigenvectors. The memory requirement therefore is modest also when N is fairly large. However, typically this approach requires more matrix-vector product evaluations with A than when all computed eigenvectors are stored.

We may compute more and more eigenpairs of A until N is such that

$$|S_m^{(N)}| = m. \tag{2.3.10}$$

This stopping criterion is referred to as the *strong convergence condition*. By Corollary 2.3.3, the set $S_m^{(N)}$ contains the indices of the m nodes with the largest f-subgraph centrality.

The criterion (2.3.10) for choosing N is useful if the required value of N is not too large. We introduce the *weak convergence criterion* to be used for problems for which the large size of N required to satisfy (2.3.10) makes it impractical to compute the associated bounds (2.3.5) with $\mathbf{u} = \mathbf{e}_i$. The weak convergence criterion is well suited for use with the hybrid algorithm for determining the most important nodes that we will describe in the following. This criterion is designed to stop increasing N when the lower bounds $L_{ii}^{(N)}$ do not increase significantly with N. Specifically, we stop increasing N, when the average increment of the lower bounds $L_{ii}^{(N)}$, $1 \le i \le n$, is small when including the Nth eigenpair $\{\lambda_N, \mathbf{v}_N\}$ in the bounds. The average contribution of this eigenpair to the bounds $L_{ii}^{(N)}$, $1 \le i \le n$, is

$$\frac{1}{n}\sum_{i=1}^{n} f(\lambda_N) v_{i,N}^2 = \frac{1}{n} f(\lambda_N),$$

and we stop increasing N when

$$\frac{1}{n} f(\lambda_N) \leq \tau \cdot \mathcal{L}_m^{(N)} \tag{2.3.11}$$

for a user-specified tolerance τ. We use $\tau = 10^{-3}$ in the computed examples. Note that when this criterion is satisfied, but not (2.3.10), the nodes with index in $S_m^{(N)}$ and with the largest lowest bounds $L_{ii}^{(N)}$ are not guaranteed to be the nodes with the largest f-subgraph centrality.

The weak convergence criterion (2.3.11) may yield a set $S_m^{(N)}$ with many more indices than m. In particular, we may not want to compute accurate bounds for the f-subgraph centrality using the approach of Section 1.5 for all nodes with index in $S_m^{(N)}$. We therefore describe how to determine a smaller index set $\mathcal{J}$, which is likely to contain the indices of the m nodes with the largest f-subgraph centrality. Since $L_{ii}^{(N)}$ generally is a better approximation of $[f(A)]_{ii}$ than $U_{ii}^{(N)}$ (cf. Remark 2.3.1), we discard from the set $S_m^{(N)}$ indices for which $L_{ii}^{(N)}$ is much smaller than $\mathcal{L}_m^{(N)}$. Thus, for a user-chosen parameter $\rho > 0$, we include in the set $\mathcal{J}$ all indices $i \in S_m^{(N)}$ such that

$$\mathcal{L}_m^{(N)} - L_{ii}^{(N)} < \rho \cdot \mathcal{L}_m^{(N)}. \tag{2.3.12}$$

In the computed examples, we use $\rho = 10^{-1}$.

The following algorithm describes the determination of the dimension N of the leading subspace and of the index set $S_m^{(N)}$. The function f is the exponential function (2.2.2). This function may be replaced by some other nonnegative nondecreasing function such as (2.2.3). The algorithm requires the adjacency matrix A, its order n, and the number of nodes with largest f-subgraph centrality desired, m. We remark that the adjacency matrix A does not have to be explicitly stored, only a function for evaluating matrix-block-vector products with A is required. In addition, the following parameters have to be provided:

- $N_{\max}$, maximum number of iterations performed;
- q, number of eigenvalues computed at each `irbleigs` call;
- $M_{\max}$, maximum number of eigenvector kept in memory;
- τ, tolerance used to detect *weak* convergence;
- ρ, tolerance used to constructed an extended list of nodes in case of weak convergence; cf. (2.3.12).

We comment on the computations of part one of the algorithm below.

Algorithm 1 first initializes the vectors $\boldsymbol{\ell}$, $\mathbf{u}$, and $\mathbf{s}$, whose components, at each iteration N, are given by

$$\ell_i = L_{ii}^{(N)}, \quad u_i = U_{ii}^{(N)}, \quad s_i = \sum_{k=1}^{N} v_{ik}^2.$$

Then, `irbleigs` is called to compute the first batch of q eigenpairs and the main loop is entered. The Boolean variable "flag" is used to detect either strong or weak convergence. The parameter $N_{\max}$ specifies the maximum number of iterations, i.e., the maximum number of times the loop made up of lines 7–29 is executed. We found it beneficial to introduce the auxiliary parameter $\overline{N}$ to keep track of how many eigenvectors from the current batch are being used. When $\overline{N} = q$, a new batch of eigenpairs is computed.

The bounds (2.3.5) with $\mathbf{u} = \mathbf{e}_i$ are computed in lines 8–13; the vector of indices $\boldsymbol{\sigma}$ contains a permutation which yields the lower bounds ℓ_i in decreasing order. In line 16 the set $S_m^{(N)}$ is constructed; subsequently the exit condition is checked. Lines 18–28 give a new batch of eigenpairs. Either one of the two kinds of restarts discussed above may be applied. If N is smaller than $M_{\max}$, then we compute the next q eigenpairs and orthogonalize the new eigenvectors against the available eigenspace. If, instead, $N \geq M_{\max}$, then we seek to determine the q eigenvalues close to the smallest available eigenvalue λ_N, and we select those that are smaller than λ_N.

Algorithm 2 describes the continued computations. The computations of the algorithm are commented on below. Lines 30–42 of the algorithm determine whether weak or strong convergence has been achieved. The variable “info” contains this information. A list of the desired nodes $\mathcal{N}$ is formed in lines 43–46, where the subgraph centralities also are updated, keeping in mind the spectrum shift at line 9 of Algorithm

Algorithm 1 Low-rank approximation, part 1

```
1:  **Input:** matrix $A$ of size $n$, number $m$ of nodes to be identified,
2:             tuning constants: $N_{\max}$, $q$, $M_{\max}$, $\tau$, $\rho$
3:  **for** $i = 1$ **to** $n$ **do** $\ell_i, s_i = 0$ **end**
4:  **call irbleigs** to compute $\{\lambda_i, \mathbf{v}_i\}_{i=1}^{q}$ such that $\lambda_i \geq \lambda_{i+1}$
5:  **if** spectrum shift is active **then** $\mu = \lambda_1$ **else** $\mu = 0$ **end**
6:  $N = 0$, $\overline{N} = 0$, flag = **true**
7:  **while** flag **and** $(N < \min\{N_{\max}, n\})$ **and** $(\overline{N} < q)$
8:      $N = N + 1$, $\overline{N} = \overline{N} + 1$
9:      $f_\lambda = \exp(\lambda_N - \mu)$
10:     **for** $i = 1$ **to** $n$ **do** $w_i = v_{iN}^2$ **end**
11:     $\mathbf{s} = \mathbf{s} + \mathbf{w}$
12:     $\boldsymbol{\ell} = \boldsymbol{\ell} + f_\lambda \cdot \mathbf{w}$
13:     $\mathbf{u} = \boldsymbol{\ell} + f_\lambda(1 - \mathbf{s})$
14:     let $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_n]$ be an index permutation such that $\ell_{\sigma_i} \geq \ell_{\sigma_{i+1}}$
15:     $L_{\max} = \ell_{\sigma_m}$
16:     $S^{(N)} = \{i : u_i \geq L_{\max}\}$
17:     flag = $(|S^{(N)}| > m)$ **and** $(\frac{1}{n} f_\lambda > \tau \cdot L_{\max})$
18:     **if** $\overline{N} = q$
19:         **if** $N < M_{\max}$
20:             **call irbleigs** to compute $\{\lambda_{N+i}, \mathbf{v}_{N+i}\}_{i=1}^{q}$ such that $\lambda_{N+i} \geq \lambda_{N+i+1}$
21:             $\overline{N} = 0$
22:         **else**
23:             **call irbleigs** to compute e-values $\nu_1 \geq \nu_2 \geq \cdots \geq \nu_q$ closest to $\lambda_N$
24:             $r = \arg\min_i |\nu_i - \lambda_N|$
25:             $\lambda_{N+i} = \nu_{r+i}$, $i = 1, \ldots, q - r$; $\mathbf{v}_{N+i}$ are associated eigenvectors
26:             $\overline{N} = r$
27:         **end if**
28:     **end if**
29: **end while**
```

1. We remark that the MATLAB implementation of Algorithm 2 contains some features not described in the algorithm. For instance, we only apply the correction due to the spectrum shift when this does not cause overflow.

Algorithm 2 Low-rank approximation, part 2

```
30: if flag
31:     $j = |S^{(N)}| - m$
32:     info = 2    % no convergence
33: else
34:     if $|S^{(N)}| > m$
35:         $\mathcal{J} = \{i : i > m,\ L_{\max} - \ell_{\sigma_i} < \rho \cdot L_{\max}\}$
36:         $j = \min(|\mathcal{J}|, 100)$, $j = \max(j, 5)$
37:         info = 1    % weak convergence
38:     else
39:         $j = 0$
40:         info = 0    % strong convergence
41:     end if
42: end if
43: for $i = 1$ to $m + j$
44:     $\mathcal{N}_i = \sigma_i$
45:     $\mathcal{V}_i = e^{\mu} \cdot \ell_{\sigma_i}$
46: end if
47: Output: list of nodes $\mathcal{N}$, subgraph centralities $\mathcal{V}$,
48:         spectrum shift $\mu$, iterations $N$, info
```

The hybrid method The hybrid method first computes a partial spectral factorization of the adjacency matrix A and applies it to determine which nodes might be the most interesting ones with respect to the criterion chosen. The discussion above exposes the situation when we would like to determine the nodes of a large network with the largest f-subgraph centrality. We also may be interested in determining the nodes with the largest f-communicability. The partial spectral factorization helps us find a set of candidate nodes that contains the nodes of interest. More accurate upper and lower bounds for expressions of the form (2.3.1) for the candidate nodes are then computed with the aid of Gauss quadrature rules. For example, this approach allows us to compute the node with the largest f-subgraph centrality of a large graph without evaluating pairs of Gauss and Gauss-Radau rules for every node of the network, i.e., for every diagonal entry of the adjacency matrix. The evaluation of Gauss-type rules for every diagonal entry can be expensive for large graphs. The computations with the hybrid method typically are considerably cheaper. This is illustrated in the following Subsection. Similarly, if we are interested in finding the nodes with the largest f-communicability, then we use (1.5.35) and compute more accurate upper and lower bounds for the candidate nodes with Gauss and Gauss-Radau rules.

2.3.2 Numerical experiments

This section presents a few examples that illustrate the performance of the methods presented in this thesis. All computations were conducted in MATLAB version 7.11

(R2010b) 64-bit for Linux, in double precision arithmetic, on an Intel Core i7-860 computer, with 8 Gb RAM. The function f is the exponential function (2.2.2).

The examples fall into three categories. The first set consists of synthetic networks which recently have been used by many authors to test the performance of computational methods. These examples are not based on real data, but reproduce typical properties of complex networks, e.g., small world effect, power law degree distribution, etc.; see [123]. They allow us to choose the dimension of the network as well as the probability value upon which they depend. The second set consists of five networks that arise in real-world applications and have publicly available data sets. Finally, we analyze a new application, which is gaining an increasing interest in software engineering, namely networks describing the dependency between modules in large software projects.

Synthetic networks In the initial examples, we consider nine classes of random symmetric adjacency matrices that were generated with the functions erdrey, geo, gilbert, kleinberg, lockandkey, pref, renga, smallw, and sticky in the MATLAB package CONTEST by Taylor and Higham [123]. For instance, the function geo [111] generates a random symmetric adjacency matrix A as follows: after determining n randomly distributed points $\mathbf{x}_i$ on the unit square, an edge is inserted between nodes i and j if $\|\mathbf{x}_i - \mathbf{x}_j\| < r$, where by default $r = \sqrt{1.44/n}$; see [123] for details on the adjacency matrices produced by the functions. The parameters for each of these functions are chosen to be their default values. Figure 2.3 shows the sparsity pattern of a typical adjacency matrix of order 1000×1000 for each kind of graph generated, as well as the number of nonzero entries of each matrix. In all computed examples, we choose the parameters $q = 20$, $N_{\max} = 300$, and $M_{\max} = 200$ for Algorithms 1 and 2.

Our first experiment is concerned with determining the node of a graph with the largest subgraph centrality, or a small set of nodes containing this node. We refer to the node with the largest subgraph centrality as the most important node. For each type of adjacency matrix, we used the bounds (2.3.5) with $\mathbf{u} = \mathbf{e}_i$ for $1 \leq i \leq n$ to determine sets of $M = 2^k$ nodes that contain the most important node of the graph for $k = 0, 1 \ldots, 9$. Table 2.1 reports, for each type of adjacency matrix and for each value of M, the average and standard deviation of the number of eigenpairs required to achieve $|S^{(N)}| \leq M$ for a sample of 100 randomly generated adjacency matrices of each kind.

For the matrix classes lockandkey, pref, renga, and sticky, the most important node in the network is correctly identified using only a fairly small number of eigenpairs; on average 12, 2, 25, and 4, respectively. For this kind of adjacency matrix, the approach of Section 2.3.1 provides a powerful method for determining the most important node or set of nodes in a large graph and for estimating subgraph centralities. The method also can be used to bound communicabilities of interest.

Turning to the adjacency matrices for graphs of the type geo, the partial spectral factorization method was able to determine a set of four nodes that contained the most important node using a modest number of eigenpairs, on the average 37 eigenpairs were needed. However, successful identification of the most important node required on average the much larger number of 228 eigenpairs. We made similar observations for adjacency matrices of the types erdrey, gilbert, kleinberg, and smallw. This illustrates that the low-rank approximation approach of Section 2.3.1 might not be able to identify the most important node(s) with a fairly small computational effort. However, the low-rank approximation approach may be useful for generating a short list of candidate nodes, whose subgraph centralities can be accurately determined by Gauss quadrature.

Table 2.2 shows the performance of the low-rank approximation method (Algo-

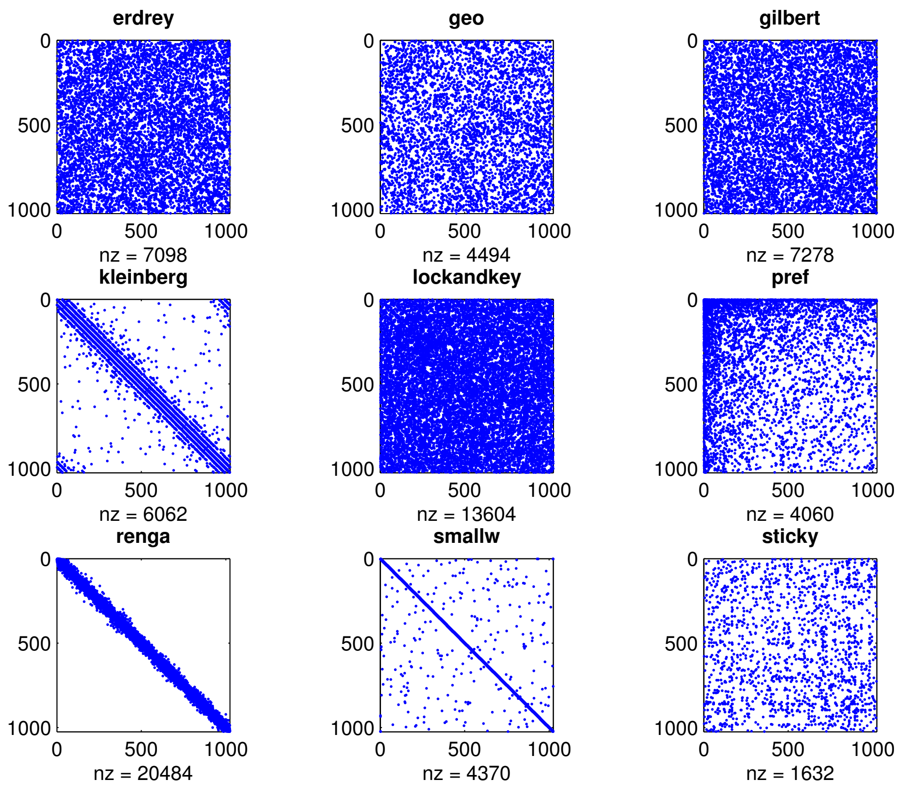


Figure 2.3: The sparsity pattern of a typical 1000×1000 matrix for each one of the nine kinds of random adjacency matrices used in the computed examples. The parameter nz shows the number of nonzero entries of each matrix.

rithms 1 and 2) when applied to test matrices of order $n = 4096$ from CONTEST. Each type of matrix was randomly generated 10 times. We wanted to identify the five most important nodes in the correct order. In the first set of experiments, we terminated the computations when the strong convergence criterion, $|S_5^{(N)}| = 5$, was satisfied. By Corollary 2.3.3, the five indices of $S_5^{(N)}$ are for the five nodes with largest f-subgraph centrality. The column "fail" reports the number of times the relative size of the lower bounds $L_{ii}^{(N)}$ does not convey the correct relative size of the f-subgraph centrality of the nodes. Table 2.2 also shows the number N of eigenpairs required to satisfy the strong convergence criterion and the computation time required. Both the value of N and the computation time are averages over 10 runs with each kind of matrix. The timings are for irbleigs with convergence tolerance $1 \cdot 10^{-3}$. This tolerance is used in all computations for this section.

The last three columns of Table 2.2 are obtained when terminating the low-rank approximation method with the weak convergence criterion (2.3.11). The columns display, in order, the number of eigenpairs computed, the execution time, and the number of candidate nodes included in the resulting list $\mathcal{N}$. The table shows that for many of the graphs, a large number of eigenpairs is required to satisfy the strong

Table 2.1: Mean and standard deviation (sdev) for nine classes of random adjacency matrices of order $n = 1024$ of the number N of eigenpairs required to achieve $|S^{(N)}| \leq M$ for $M = 2^k,\ 0 \leq k \leq 9$. The sample size is 100 matrices of each kind.

test matrix	M	512	256	128	64	32	16	8	4	2	1
erdrey	mean	65	69	75	82	92	104	121	147	187	261
	sdev	12	13	15	18	21	25	32	45	63	128
geo	mean	11	11	11	12	13	17	24	37	97	228
	sdev	6	6	6	7	9	12	14	22	194	350
gilbert	mean	65	69	75	83	93	106	125	151	200	287
	sdev	13	14	16	19	23	29	40	52	92	170
kleinberg	mean	113	121	130	140	152	167	185	212	263	337
	sdev	10	11	14	17	20	27	35	49	94	139
lockandkey	mean	2	2	2	2	2	2	2	2	3	12
	sdev	0	0	0	0	0	0	0	1	4	27
pref	mean	2	2	2	2	2	2	2	2	2	2
	sdev	0	0	0	0	0	0	0	0	0	2
renga	mean	17	17	18	18	19	20	20	21	23	25
	sdev	1	1	1	1	1	1	1	2	2	4
smallw	mean	224	227	243	245	248	266	312	342	406	607
	sdev	18	19	25	25	26	47	98	134	196	273
sticky	mean	2	2	2	2	2	2	2	2	3	4
	sdev	1	1	1	1	1	1	1	2	3	6

convergence criterion, while the low-rank method for all graphs succeeds in determining a small list of candidate nodes in fairly short time, with the possible exception of the smallw graphs. The latter graphs are particularly difficult for the low-rank method. A possible reason is the fact that almost all the nodes of a typical smallw network have quantized subgraph centralities, in the sense that each value is very close to another one among a small number of possible values. This leads to that a large number N of eigenpairs are required to satisfy the strong or weak convergence criteria. This is illustrated in Figure 2.4, which displays the eigenvalues and the subgraph centralities for the nodes of three test networks with 512 nodes each. It is immediate to observe that the eigenvalue decay alone is not sufficient to predict the speed of convergence of the algorithm.

Columns 2 and 3 of Table 2.3 report the number of matrix-vector product evaluations and the computational time (in seconds) required when determining the five nodes with the largest f-subgraph centrality for graphs with $n = 4096$ nodes of the type indicated, by evaluating pairs of Gauss and Gauss-Radau quadrature rules as described in Section 1.5. A matrix-vector product with a block vector with k columns is counted as k matrix-vector product evaluations. The performance of the hybrid method described in Section 2.3.1 is shown in columns 4 and 5. The hybrid method can be seen to require fewer matrix-vector product evaluations and shorter execution time. The number of matrix-vector products and the execution times are averages over 10 runs with randomly generated graphs of the appropriate kind. Table 2.3 compares these approaches with two classical methods, namely the evaluation of the matrix exponential using the MATLAB function `expm`, which is based on Padé approximation, and complete spectral factorization with spectrum shift of the adjacency matrix using

Table 2.2: Results obtained by the low-rank approximation method with termination due to strong or weak convergence. The table shows the number of failures, the number N of eigenpairs required to satisfy the specified termination criterion, the execution time in seconds, and in case of weak convergence the cardinality of the list $\mathcal{N}$ of candidate nodes. Each test was repeated 10 times with 4096×4096 adjacency matrices.

	strong convergence			weak convergence		
test matrix	fail	N	time	N	time	$\lvert\mathcal{N}\rvert$
erdrey	3	304	4.2e+01	54	2.7e+00	10
geo	1	198	1.2e+01	4	2.3e-01	10
gilbert	4	345	5.0e+01	50	2.8e+00	10
kleinberg	9	346	4.1e+01	150	9.7e+00	11
lockandkey	0	35	3.4e+00	2	5.1e-01	8
pref	0	6	2.6e-01	2	2.8e-01	10
renga	0	112	2.0e+00	45	8.6e-01	13
smallw	9	451	7.7e+01	288	4.8e+01	1654
sticky	1	4	2.8e-01	2	2.6e-01	8

the MATLAB function `eig`. All methods compared in Table 2.3 correctly identified the five most important nodes of each network. The table shows the Gauss quadrature approach to be faster than `expm` and `eig`, and the hybrid method to be the fastest, except for the smallw network. The execution times for the Gauss quadrature and hybrid algorithms, as well as for `expm`, change significantly with the network. We note that the Gauss quadrature and hybrid algorithms require far less storage space than the `expm` function, which needs to allocate up to six matrices of the same size as the input matrix.

We also applied the hybrid method to smaller networks of the same kind as used for Table 2.3. Specifically, we generated networks with 512, 1024, and 2048 nodes. The hybrid method successfully determined the five nodes with the largest f-subgraph centrality for these graphs.

Having at our disposal a set of synthetic examples, whose size can be chosen at will, makes it possible to investigate the scalability of the methods considered. We let $n = 100, 200, \ldots, 2000$ and, for each of the CONTEST networks, we measured the execution time corresponding to the four algorithms of Table 2.3. There are essentially four classes of results, depicted in Figure 2.5. Most of the examples behave similarly to the lockandkey network: the `expm` function is convenient only for small sizes and it is generally slower than the approach based on eigenvalues (`eig`). Gauss quadrature is faster than the first two methods when n gets large enough. The hybrid method is the fastest for almost all large and small networks.

In the kleinberg example the performance of the four methods is quite similar, but still the hybrid approach is slightly faster. In the renga network the `expm` function is faster than `eig` and Gauss quadrature, but all of them are slower than `hybrid`. Similar results hold for geo. The smallw results are totally different from the others: the hybrid method behaves like `eig`, and `expm` is the fastest method.

The block Lanczos method implemented by the MATLAB function `irbleigs` requires a user to select block-size. Figure 2.6 shows the influence of the block-size k on the execution time required by the hybrid method. The figure displays timings for block-sizes $k = 1, 2, \ldots, 10$, for a particular realization of four kinds of adjacency

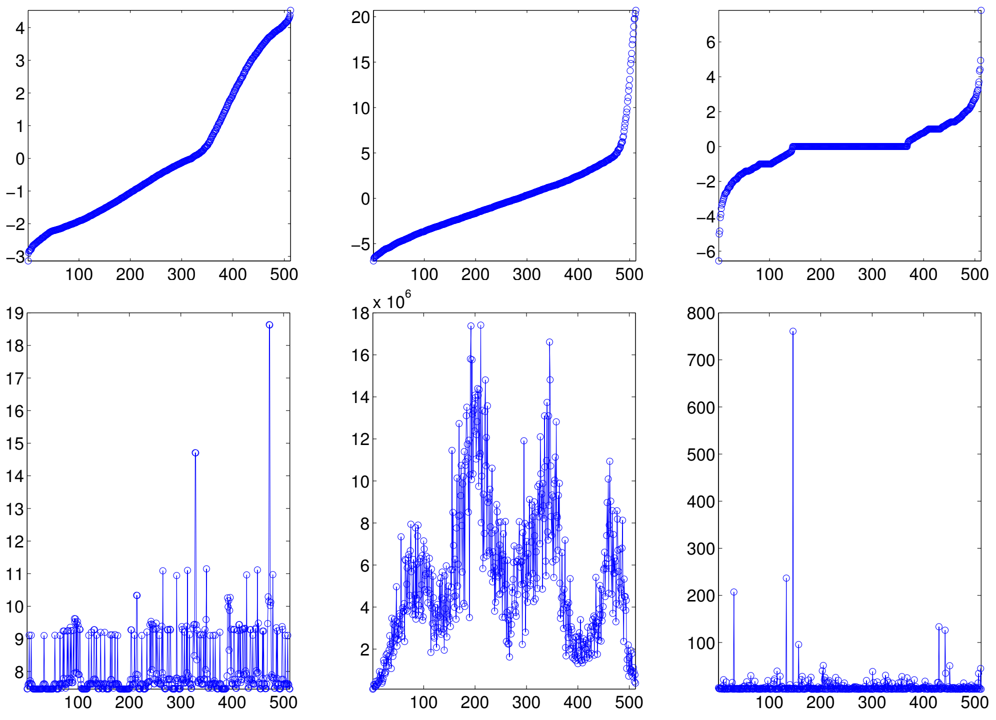


Figure 2.4: From right to left: Eigenvalue distribution (top) and subgraph centralities (bottom) for the networks smallw, renga, and sticky with $n = 512$ nodes.

matrices of order 4096. The effect of the block-size on the execution time is further illustrated in Table 2.4, where the performance of the hybrid method for block-size $k = 1$ is compared with the results in Table 2.3, obtained for block-size $k = 5$. We observed in our experiments that for most matrices the execution times were the smallest for block-size about 5 or 6, and only in a few cases the computation was faster with $k = 1$; see the matrix smallw in Figure 2.6 and Table 2.4. For this reason, we used block-size $k = 5$ in the computed examples reported above.

Real-world networks The following examples come from real-world applications. The first network (2114 nodes, 4480 edges) describes the protein interaction network for yeast: each edge represents an interaction between two proteins [81, 122]. The data set was originally included in the Notre Dame Networks Database and is now available at [109]. The eigenvalue distribution of the adjacency matrix is shown in Figure 2.7, left.

The second network (4941 nodes, 13188 edges) is an undirected unweighted representation of the topology of the western states power grid of the United States, compiled by Watts and Strogatz [133], and is now available at [98]. Figure 2.7, right, reports the eigenvalues of the adjacency matrix.

The third example is a symmetrized snapshot of the structure of the Internet at the level of autonomous systems (22963 nodes, 96872 edges), reconstructed from BGP (Border Gateway Protocol) tables posted by the University of Oregon Route Views

Table 2.3: Comparison of Gauss quadrature and the hybrid algorithm. Each test is repeated 10 times with $n = 4096$. For each kind of adjacency matrix, we report the average number of matrix-vector product evaluations required (mvp) and the average execution time in seconds.

	Gauss		Hybrid		`expm`	`eig`
test matrix	mvp	time	mvp	time	time	time
erdrey	28663	1.5e+01	4875	3.0e+00	5.5e+02	1.7e+02
geo	19913	9.6e+00	496	2.3e-01	7.6e+01	1.5e+02
gilbert	28663	1.5e+01	4558	2.7e+00	5.6e+02	1.7e+02
kleinberg	22343	1.1e+01	11710	1.0e+01	2.7e+01	1.7e+02
lockandkey	32766	2.0e+01	1054	5.6e-01	5.2e+02	1.7e+02
pref	36607	1.8e+01	587	2.7e-01	4.6e+02	1.6e+02
renga	36087	2.2e+01	1591	9.5e-01	9.7e+01	1.6e+02
smallw	19266	8.6e+00	77979	5.5e+01	9.5e+00	1.6e+02
sticky	21792	1.0e+01	650	2.9e-01	1.3e+02	1.5e+02

Table 2.4: Comparison between hybrid algorithms with block-size $k = 5$ and $k = 1$. Each test is repeated 10 times, with $n = 4096$.

	Hybrid, $k = 5$		Hybrid, $k = 1$		expm	eig
test matrix	mvp	time	mvp	time	time	time
erdrey	4875	3.0e+00	2190	1.7e+00	5.5e+02	1.7e+02
geo	496	2.3e-01	1044	7.1e-01	7.6e+01	1.5e+02
gilbert	4558	2.7e+00	2181	1.6e+00	5.6e+02	1.7e+02
kleinberg	11710	1.0e+01	2271	1.6e+00	2.7e+01	1.7e+02
lockandkey	1054	5.6e-01	2106	1.7e+00	5.2e+02	1.7e+02
pref	587	2.7e-01	856	5.6e-01	4.6e+02	1.6e+02
renga	1591	9.5e-01	2208	1.7e+00	9.7e+01	1.6e+02
smallw	77979	5.5e+01	2271	1.4e+00	9.5e+00	1.6e+02
sticky	650	2.9e-01	662	4.2e-01	1.3e+02	1.5e+02

Project. This snapshot was created by Newman from data for July 22, 2006, and is available at [98].

The fourth example is the collaboration network of scientists posting preprints on the condensed matter archive at `www.arxiv.org` [104], and consists of 40421 nodes and 351304 edges. This version is based on preprints posted to the archive between January 1, 1995 and March 31, 2005. The original network is weighted, here we consider an unweighted version. The data set is available at [98].

The fifth and largest example describes all the user-to-user links (*friendship*) from the Facebook New Orleans networks. Each edge between two users means that the second user appeared in the first user's friend list. The network (63731 nodes, 1545686 edges) was studied in [128], and the data set is available at [124]. We symmetrized the adjacency matrix since the friendship relation in Facebook is reflexive.

Table 2.5 shows, for each of the above networks, the results obtained by the low rank approximation, either with strong or weak convergence test, when identifying the five most important nodes. In the two smaller examples the strong convergence

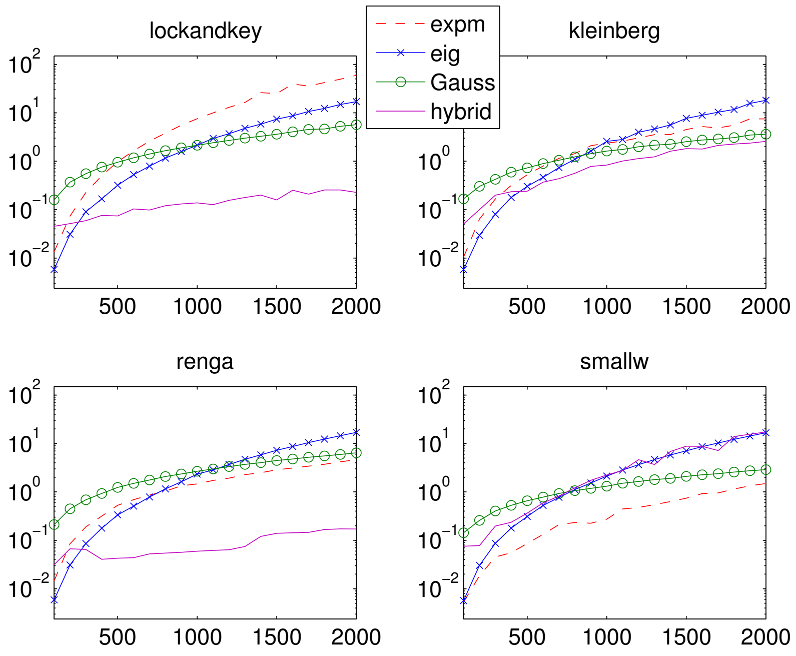


Figure 2.5: Execution time for the algorithms considered when the size of the network varies.

criterion requires a large number of eigenpairs, but the weak convergence test is satisfied when the candidate list contains just 10 nodes. On the contrary, only 2 eigenpairs suffice to identify the five most important nodes of the three largest networks. We do not specify a value in the column labeled “fail” of Table 2.5 for the largest networks, because we cannot evaluate `expm` in reasonable time when $n > 10^4$ and compare with the ordering obtained when using this function. However, we note that both the strong and weak convergence of Algorithms 1 and 2, as well as Gauss quadrature and the hybrid algorithms, produce the same five nodes in the same order.

In Table 2.6, the number of matrix-vector product evaluations and the execution time corresponding to Gauss quadrature, the hybrid method, `expm`, and `eig`, are compared. The computations required for the Gauss quadrature method and for the MATLAB functions `expm` and `eig` can be seen to be quite time-consuming for large matrix sizes. Results for `expm` and `eig` for the largest networks therefore are not reported when $n > 10^4$. The results achieved with the hybrid method are very encouraging and illustrate the possibility of applying hybrid methods to large-scale problems.

Software networks One of the main issues in modern software engineering is to apply certain metrics to software packages in order to gain information about their properties. Concas et al. [27, 28] associate a graph to a software package and study the connection between properties of the graph and the occurrence of bugs during the package development. We considered the software package Netbeans (`http://www.netbeans.org/`), an integrated development environment. This choice

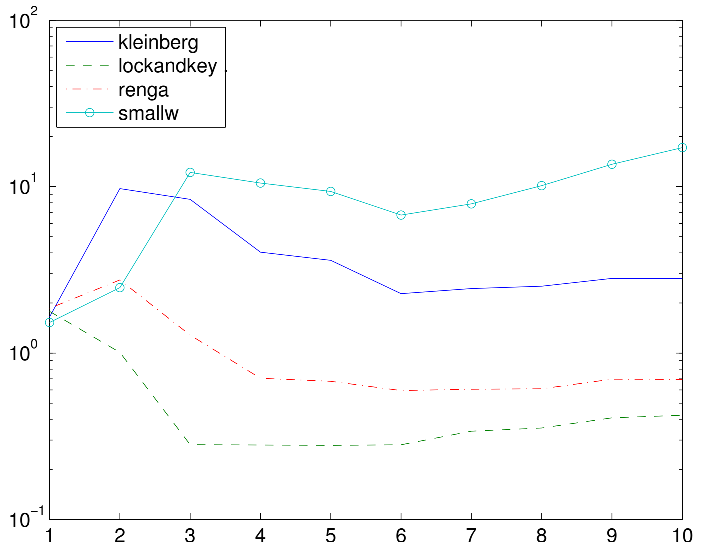


Figure 2.6: Hybrid algorithm: execution time versus block-size for four classes of adjacency matrices, $n = 4096$.

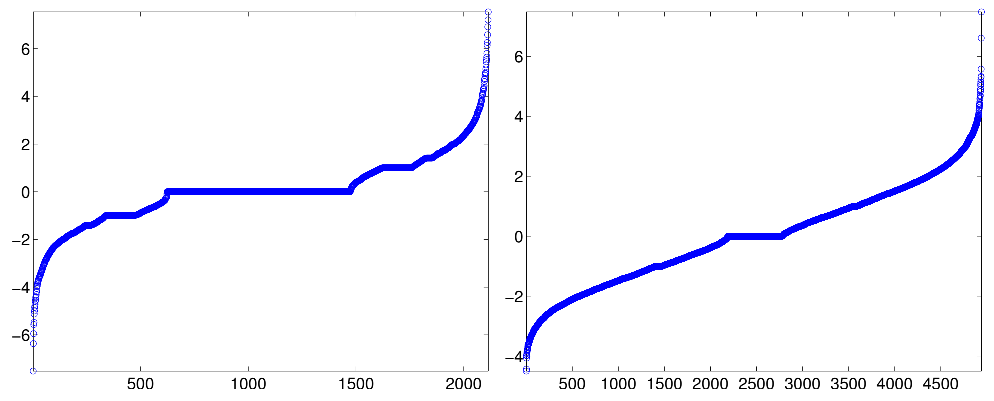


Figure 2.7: Eigenvalues of the adjacency matrix of the yeast (left) and power (right) networks.

is motivated by the fact that for this package, the source code for several versions is available and so is complete information about the localization of bugs in different versions and software modules. The Netbeans system is written in Java and its classes are contained in source files referred to as compilation units (CU). A CU generally contains just one class, but may occasionally contain two or more classes. A software network is a graph, in which the CUs are nodes and the directed edges correspond to a CU referencing another one. The resulting network is directed, not connected, and has self-edges, i.e., there may be edges connecting a node to itself. The network is quite large, with 44581 nodes and 189646 links.

Following [27, 28], we symmetrize the adjacency matrix by considering each edge without orientation. This allows us to test the performance of the algorithms discussed

Table 2.5: Results obtained by the low-rank approximation algorithm with both strong and weak convergence criteria. The table reports the number of failures, the number N of eigenpairs required to reach convergence, the execution time, and, in case of weak convergence, the cardinality of the list $\mathcal{N}$ of candidate nodes.

			strong convergence			weak convergence		
network	nodes	edges	fail	N	time	N	time	$\lvert\mathcal{N}\rvert$
yeast	2114	4480	0	66	1.5e+00	7	1.2e-01	10
power	4941	13188	0	230	3.4e+01	3	5.8e-01	10
internet	22963	96872	–	2	1.6e+00	2	1.7e+00	5
collaborations	40421	351304	–	2	3.8e+00	2	3.6e+00	5
facebook	63731	1545686	–	2	1.5e+01	2	1.4e+01	5

Table 2.6: Comparison of Gauss quadrature and hybrid algorithms. For each matrix, we report the number of matrix-vector product evaluations (mvp) and the execution time in seconds.

		Gauss		Hybrid		expm	eig
network	nodes	mvp	time	mvp	time	time	time
yeast	2114	9028	3.0e+00	367	1.4e-01	1.2e+01	1.5e+01
power	4941	23317	1.5e+01	759	6.4e-01	3.3e+01	2.6e+02
internet	22963	236345	3.8e+02	679	2.4e+00	–	–
collaborations	40421	431146	1.2e+03	835	4.5e+00	–	–
facebook	63731	801960	7.9e+03	859	1.2e+01	–	–

above, and to perform an initial analysis of the network. The sparsity pattern of the original Netbeans network and of the symmetrized version is displayed in Figure 2.8.

Tables 2.7 and 2.8 present results for several subnetworks of Netbeans as well as for the entire network. Each subnetwork corresponds to a subsystem of the whole software package. Our task was to determine the five most important nodes of the entire network as well as of each subnetwork. It is quite remarkable that only 2 to 5 iterations with Algorithms 1 and 2 suffice to satisfy the strong convergence criterion and only 2 iterations are needed to satisfy the weak convergence criterion. It is also interesting that Algorithms 1 and 2 require about the same number of iterations N for each subnetwork as well as for the entire network. As explained above, we did not apply the `expm` and `eig` functions to matrices of order larger than 10^4.

Using available information about bugs in the CUs and the computed values for the f-subgraph centrality and starting convenience gives Table 2.9. The entries of the top table represent the number of CUs without and with bugs, with subgraph centrality either smaller or larger than the average. The bottom table reports similar information for the starting convenience. The tables yield the chi-square test statistic $\chi^2 = 2997$ and $\chi^2 = 3503$, respectively. Both values are much larger than the critical value $\chi^2_\alpha = 7.88$ from the chi-square distribution for $\alpha = 5 \cdot 10^{-3}$. Since we are testing the hypothesis of stochastic independence, we are led to accept, with probability 99.5%, the hypothesis that the presence of bugs in a CU depends on both the subgraph centrality and the starting convenience. The results of Table 2.9 suggests that there is a low probability to find bugs in CUs characterized by a low value of the computed

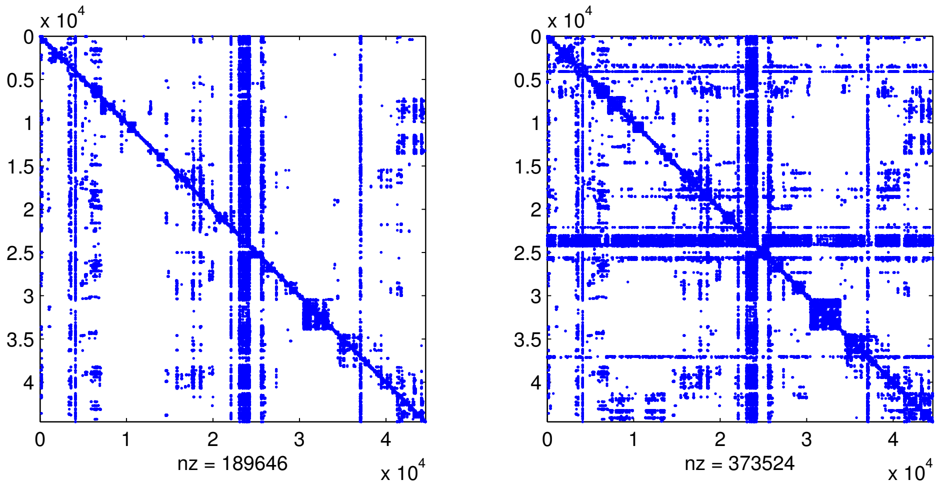


Figure 2.8: The sparsity pattern of the Netbeans network: original adjacency matrix (left), symmetrized version (right). The parameter nz shows the number of nonzero entries of each matrix.

metrics. This example illustrates that the hybrid method can be applied to large-scale problems. Both the storage and execution time of the method are fairly small.

Application of block algorithms The main advantage of the approach described above is that it yields bounds for (2.3.1). However, the computational effort required may be much larger than when block methods are used. Suppose that bounds are desired for each element of the leading $k \times k$ submatrix of $f(A)$, i.e., of

$$\mathcal{I}f = W^T f(A) W$$

with $W = [\boldsymbol{e}_1, \dots, \boldsymbol{e}_k]$. Using the approach of this section requires $k(k+1)/2$ partial Lanczos decompositions with $\boldsymbol{w} = \boldsymbol{e}_i$ for $1 \leq i \leq k$ and $\boldsymbol{w} = (\boldsymbol{e}_i + \boldsymbol{e}_j)/\sqrt{2}$ for $1 \leq i < j \leq k$. Approximations of these bounds can be computed by the block methods

Table 2.7: Results obtained by the low-rank approximation algorithm with both strong and weak convergence criteria. The table reports the number of failures, the number N of eigenpairs required to reach convergence, the execution time, and, in case of weak convergence, the cardinality of the list $\mathcal{N}$ of candidate nodes.

			strong convergence			weak convergence		
network	nodes	edges	fail	N	time	N	time	$\lvert\mathcal{N}\rvert$
profiler	1231	4161	0	2	3.3e-01	2	1.1e-01	5
j2ee	2100	4642	0	3	1.1e-01	2	1.5e-01	10
uml	3462	13130	0	2	1.6e-01	2	2.1e-01	5
enterprise	13548	15604	–	5	6.5e-01	2	7.3e-01	10
netbeans	44581	189646	–	2	4.9e+00	2	4.3e+00	5

Table 2.8: Comparison of Gauss quadrature and hybrid algorithms. For each matrix, we report the number of matrix-vector product evaluations (mvp) and the execution time in seconds.

		Gauss		Hybrid		expm	eig
network	nodes	mvp	time	mvp	time	time	time
profiler	1231	10265	3.0e+00	429	1.1e-01	1.3e+01	3.7e+00
j2ee	2100	15654	4.9e+00	386	1.1e-01	2.7e+01	1.6e+01
uml	3462	33541	1.7e+01	655	2.7e-01	2.8e+02	8.1e+01
enterprise	13548	55136	4.1e+01	640	8.3e-01	–	–
netbeans	44581	450078	1.3e+03	679	3.7e+00	–	–

Table 2.9: Contingency table reporting the frequency of bugs with respects to either subgraph centrality or starting convenience.

	subgraph centrality	
bugs	smaller than average	larger than average
no	33563	4501
yes	4004	2513

	starting convenience	
bugs	smaller than average	larger than average
no	30028	8035
yes	3053	3465

described in Section 1.4.5. They require the evaluation of a single partial block Lanczos decomposition with block-size k. If the Lanczos method with block-size one requires about the same number of steps as the block Lanczos method with block-size k, then the count of matrix-vector products scales with k for the block method, while it scales with k^2 when the block-size is one. Here we count the product of the matrix A with a block-vector with k columns as k matrix-vector products.

In the following, we illustrate the application of block Gauss and anti-Gauss quadrature rules to the estimation of certain functions of symmetric or nonsymmetric adjacency matrices. All computations were carried out with MATLAB version 8.1 (R2013a) 64-bit for Linux, in double precision arithmetic, on an Intel Core i7-860 computer with 8 GB RAM. The following numerical examples illustrate the performance of block Gauss and block anti-Gauss quadrature rules when applied to integrate the exponential function. The matrices for most examples are adjacency matrices for undirected networks that arise in real-world applications and are publicly available.

We will approximate $\mathcal{I}f$ by

$$F_N := \mathcal{A}_{N+1}f = \frac{1}{2}\left(\mathcal{G}_N f + \mathcal{H}_{N+1}f\right); \tag{2.3.13}$$

cf. (1.5.21). Seeking to determine each entry of $\mathcal{I}f$ with an approximate relative tolerance τ, we terminate the symmetric block Lanczos methods at iteration N, where

N is the smallest integer such that

$$T_N := \frac{1}{2}\frac{\|\mathcal{G}_N f - \mathcal{H}_{N+1} f\|_{\max}}{\|F_N\|_{\max}} < \tau, \tag{2.3.14}$$

with $\|B\|_{\max} := \max_{1\le i,j\le k} |B_{ij}|$, and then accept (2.3.13) as our approximation of $\mathcal{I}f$. In all experiments of this section, we let $\tau = 10^{-3}$.

Assume that

$$\min\{[\mathcal{G}_N f]_{ij}, [\mathcal{H}_{N+1} f]_{ij}\} \le [\mathcal{I}f]_{ij} \le \max\{[\mathcal{G}_N f]_{ij}, [\mathcal{H}_{N+1} f]_{ij}\}.$$

Then

$$\begin{aligned} |[F_N - \mathcal{I}f]_{ij}| &= \left| \left[\frac{1}{2}\mathcal{G}_N f - \frac{1}{2}\mathcal{I}f\right]_{ij} + \left[\frac{1}{2}\mathcal{H}_{N+1} f - \frac{1}{2}\mathcal{I}f\right]_{ij} \right| \\ &\le \frac{1}{2}|[\mathcal{G}_N f - \mathcal{I}f]_{ij}| + \frac{1}{2}|[\mathcal{H}_{N+1} f - \mathcal{I}f]_{ij}| \\ &= \frac{1}{2}|[\mathcal{G}_N f - \mathcal{H}_{N+1} f]_{ij}|. \end{aligned}$$

Therefore, if $[\mathcal{G}_N f]_{ij}$ and $[\mathcal{H}_{N+1} f]_{ij}$ bracket $[\mathcal{I}f]_{ij}$ for all $1 \le i, j \le k$, then (2.3.14) implies that

$$G_N := \frac{\|F_N - \mathcal{I}f\|_{\max}}{\|F_N\|_{\max}} < \tau, \tag{2.3.15}$$

i.e., F_N approximates $\mathcal{I}f$ elementwise with a relative error bounded by about τ.

Computations with the function $f(A) = \exp(A)$ may give rise to overflow when the adjacency matrix A is large. This can be avoided by spectrum shift in the following way. Consider the case of a symmetric block tridiagonal matrix J_N. When evaluating (1.5.12), the shift μ is chosen to be the largest eigenvalue of J_N, and we evaluate the exponential of the shifted matrix $J_N - \mu I_{kN}$ using the spectral factorization of J_N. The factor $\exp(\mu)$ does not have to be computed. The computation of (1.5.24) is carried out similarly.

We have not experienced breakdown in any of the reported computations. However, breakdowns have been observed in some examples when the block size is larger than 5.

We present some examples that show the performance of block Gauss and anti-Gauss quadrature rules associated with the symmetric block Lanczos method. This method is applied to seven real-world undirected unweighted networks, some of which have been investigated numerically in [51]. The networks have the following properties:

Email (1133 nodes, 10902 edges) is a representation of e-mail interchanges between members of the University Rovira i Virgili (Tarragona), described in [65]. The data set is available at Alex Arena's web page [1].

Autobahn (1168 nodes, 2486 edges) describes the German highway system network. The nodes are German locations and the edges are highways connecting them. It is available at [14].

Yeast (2114 nodes, 4480 edges) describes the protein interaction network for yeast. Each edge represents an interaction between two proteins [81, 122]. The data set was originally included in the Notre Dame Networks Database, and it is now available at [109].

Power (4941 nodes, 13188 edges) is an undirected unweighted representation of the topology of the western states power grid of the United States, compiled by Watts and Strogatz [133]. The original data set was made available at the web site of Watts at Columbia University, and now can be found at [98].

Internet (22963 nodes, 96872 edges) is a symmetrized snapshot of the structure of the Internet at the level of autonomous systems, reconstructed from BGP (Border Gateway Protocol) tables posted by the University of Oregon Route Views Project. This snapshot was created by Mark Newman from data for July 22, 2006 [98].

Collaboration (40421 nodes, 351304 edges) is the collaboration network of scientists who submitted preprints to the condensed matter archive at www.arxiv.org [104] between January 1, 1995, and March 31, 2005. The original network is weighted, here we consider an unweighted version [98].

Facebook (63731 nodes, 1545686 edges) is the largest example we consider. It describes all the user-to-user links (*friendships*) of the Facebook New Orleans network. It was studied in [128], and the data set is available at [124].

We are interested in computing the f-subgraph centrality of k specified nodes, as well as the f-communicability between each pair of nodes, for a total of $k(k+1)/2$ numerical quantities. We may assume that the nodes of interest are the nodes 1 through k. We, therefore, seek to approximate $\mathcal{I}f = W^T f(A) W$ with $W := [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k]$.

To investigate whether for each N the entries $[\mathcal{G}_N f]_{ij}$ and $[\mathcal{H}_{N+1} f]_{ij}$ bracket the quantity $[\mathcal{I}f]_{ij}$, we applied block Gauss and anti-Gauss rules to compute the centrality and communicability for five nodes of the first four networks. These networks are small enough to allow the evaluation of the matrix exponential by the MATLAB function `expm`. Since we cannot assume that the values returned by `expm` are exact, we checked that they are *almost* bounded by the computed quantities, i.e., whether

$$\mathcal{L} - \sqrt{\epsilon_M} < [\texttt{expm(A)}]_{ij} < \mathcal{U} + \sqrt{\epsilon_M}, \qquad i,j = 1, \ldots, 5, \tag{2.3.16}$$

where $\mathcal{L} = \min\{[\mathcal{G}_N f]_{ij}, [\mathcal{H}_{N+1} f]_{ij}\}$, $\mathcal{U} = \max\{[\mathcal{G}_N f]_{ij}, [\mathcal{H}_{N+1} f]_{ij}\}$, and $\epsilon_M \simeq 2.2 \cdot 10^{-16}$. These inequalities were satisfied for the first four networks, i.e., for all networks for which we could verify them.

Table 2.10 shows the execution time for the block Gauss and anti-Gauss quadrature rules, the scalar Gauss/Gauss–Radau quadrature rules described in Section 1.5, and the `expm` function. We apply the quadrature rules to compute approximations of $\mathcal{I}f = W^T \exp(A) W$ with $W = [\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_4, \boldsymbol{e}_5]$ and terminate the Lanczos and block Lanczos methods as soon as $T_N < 10^{-3}$. The last networks of Table 2.10 are too large to allow the evaluation of the function `expm`. It can be seen that Lanczos methods can be applied to rather large networks, and that the block Lanczos method with block-size 5 is faster than the standard Lanczos method with block-size 1.

Table 2.11 displays the number of matrix-vector product (MVP) evaluations and the error G_N defined by (2.3.15) for both the scalar and the block case. The number of MVPs equals the number of steps of the Lanczos method when the block-size is one, and equals the product of the number of Lanczos steps and the block-size when the latter is larger than one. When evaluating G_N, we consider the value returned by `expm` to be exact. The error is only reported for the smallest networks and shows the termination criterion always to give approximations with the desired accuracy. Since

Table 2.10: Execution time, in seconds, for computing centralities of and communicabilities between five nodes of undirected networks.

Matrix	Nodes	Edges	`expm`	Block-size 1	Block-size 5
Email	1133	10902	1.2e+01	1.3e-01	3.4e-02
Autobahn	1168	2486	8.2e-01	3.1e-01	2.9e-02
Yeast	2114	4480	1.0e+01	9.6e-02	3.3e-02
Power	4941	13188	2.1e+01	1.3e-01	2.7e-02
Internet	22963	96872	–	6.9e-01	1.2e-01
Collab.	40421	351384	–	1.6e+00	3.5e-01
Facebook	63731	1634180	–	4.5e+00	6.0e-01

the last three networks are too large to allow the evaluation of `expm`, we cannot report the errors for them. To better illustrate the effect of the block-size on the execution time, we let the number of columns k of W increase. For each k we approximate the entries of $W^T f(A)W$ both by the symmetric block Lanczos method with block-size k and by $k(k+1)/2$ applications of the standard Lanczos method with block-size one. The left panel of Figure 2.9 shows execution times, and the right panel the ratio between the execution times for block-sizes 1 and k. The speed-up of the block method is roughly linear as a function of the block-size.

Table 2.11: Number of MVP evaluations and size of the error G_N (2.3.15) for both block-sizes 1 and 5.

			Block size 1		Block size 5		
Matrix	Nodes	Edges	MVP	G_N	MVP	Steps	G_N
Email	1133	10902	75	2.2e-05	40	8	2.8e-06
Autobahn	1168	2486	36	1.8e-05	25	5	6.3e-07
Yeast	2114	4480	35	8.1e-05	35	7	2.6e-06
Power	4941	13188	45	5.9e-06	30	6	4.7e-07
Internet	22963	96872	95	–	35	7	–
Collab.	40421	351384	100	–	50	10	–
Facebook	63731	1634180	102	–	50	10	–

A reason for the speed-up shown in Figure 2.9 is that both centralities and communicabilities were requested. We next investigate whether block methods are competitive when only centralities are desired. Figure 2.10 depicts execution times for this situation. The figure is analogous to Figure 2.9 and shows that for the computer used in our experiments, block-size 16 yields a speed-up of a little less than a factor of 2. Thus, also in the situation when only subgraph centralities are needed, the block method is competitive. In particular, it may be attractive to apply block Lanczos methods in the hybrid scheme for identifying the k nodes of a network with the largest subgraph centrality discussed above.

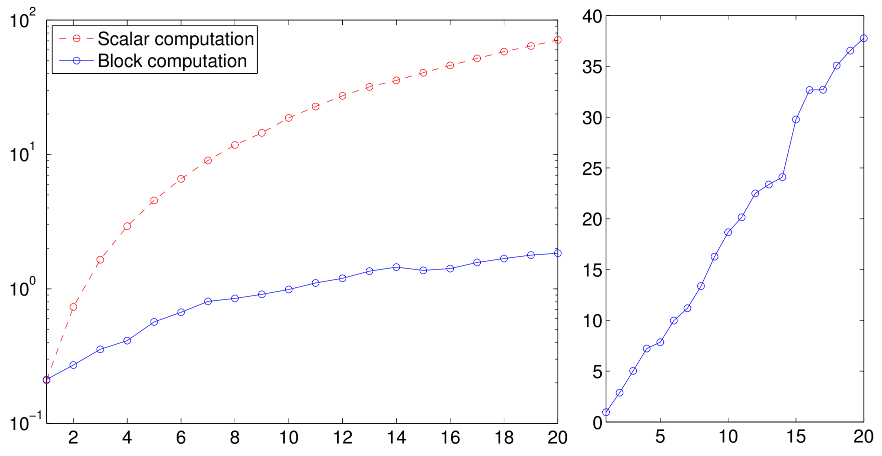


Figure 2.9: Execution times for symmetric Lanczos methods with block-size 1 (top graph) and for block-size k (bottom graph) as a function of k when determining approximations of the entries of $W^T f(A)W$. The right panel shows the ratio between the timings.

2.4 Directed networks

We are interested in studying large unweighted directed networks without multiple edges and loops. This kind of network can be described with a directed graph G. The nodes are represented by vertices of the graph and the connections between adjacent nodes by directed edges. We assume that the number of nodes, n, is large and that the number of edges is much smaller than n^2. Networks that give rise to this kind of graphs arise in many scientific and industrial applications, including genetics, epidemiology, energy distribution, and telecommunications; see, e.g., [13, 17, 30, 43, 46, 75, 103] and references therein. For instance, internet search engines use graphs that describe the connections between web pages.

For directed graphs, the size of $[\exp(A)]_{ii}$ is not always a meaningful measure of the importance of node i. Benzi et al. [11] illustrated this with the following example. Let A be a Jordan block

$$A = [A_{ij}] \in \mathbb{R}^{n\times n}, \qquad A_{ij} = \begin{cases} 1, & j = i+1, \\ 0, & j \neq i+1. \end{cases}$$

Then $[\exp(A)]_{ii} = 1$ for all i, and these values do not in an obvious manner correspond to intuition about the importance of the nodes in the associated network; for instance, in many applications it is meaningful to consider the first node less important than the remaining nodes.

To remedy this difficulty, Benzi et al. [11] proposed to consider the exponential of the matrix

$$\mathcal{A} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \qquad (2.4.1)$$

when A is a nonsymmetric adjacency matrix. Here the superscript ^{T} denotes transpo-

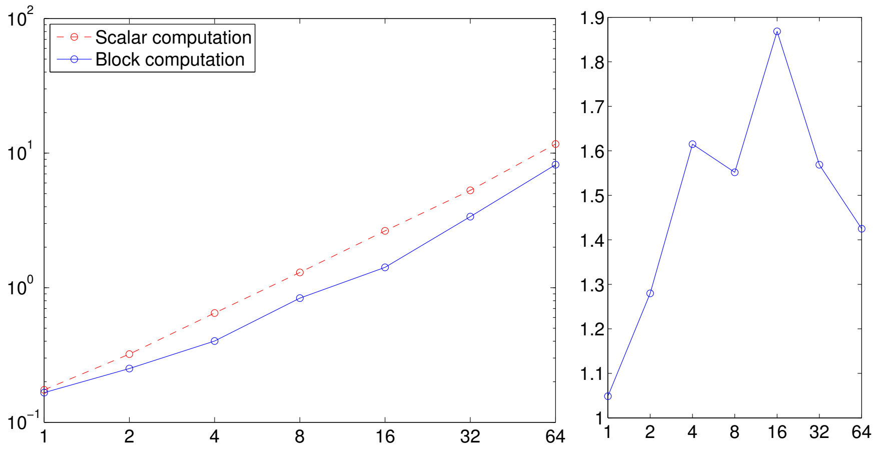


Figure 2.10: Execution times for block-size 1 (top graph) and block-size k (bottom graph) as a function of k when computing the subgraph centralities of k nodes. The right panel shows the ratio between the timings.

sition. Using the singular value decomposition (SVD) of A, it is easy to see that

$$\exp(\mathcal{A}) = \begin{bmatrix} \cosh(\sqrt{AA^T}) & A\,\mathrm{s}(\sqrt{AA^T}) \\ \mathrm{s}(\sqrt{AA^T})A^T & \cosh(\sqrt{A^TA}) \end{bmatrix}, \tag{2.4.2}$$

where

$$\mathrm{s}(t) = \begin{cases} t^{-1}\sinh(t), & t \neq 0, \\ 1, & t = 0; \end{cases} \tag{2.4.3}$$

see (2.4.7) below. Let $M^\dagger$ denote the Moore–Penrose pseudoinverse of the matrix M. Then

$$A\,\mathrm{s}(\sqrt{AA^T}) = A(\sqrt{AA^T})^\dagger \sinh(\sqrt{AA^T}).$$

The use of the matrix (2.4.1) is justified by Benzi et al. [11] by its connection to the Hypertext Induced Topics Search (HITS) algorithm by Kleinberg [82]; see also [103, Section 7.5]. Within the framework of this algorithm, there are two kinds of significant nodes: hubs and authorities. Hubs are distinguished by the fact that they point to many important nodes. The latter are referred to as authorities. Important hubs are nodes that point to many important authorities, important authorities are pointed to by important hubs; see also Blondel et al. [17] for a discussion on this and related network models. We will return to the HITS algorithm in Section 2.4.2.

The matrix entries $[(AA^T)^k]_{ij}$ and $[(A^TA)^k]_{ij}$ count the number of alternating walks of length $2k$. Following Benzi et al. [11], we refer to

$$[\exp(\mathcal{A})]_{ii} = [\cosh(\sqrt{AA^T})]_{ii}, \qquad 1 \leq i \leq n,$$

as the hub centrality of node i, and to

$$[\exp(\mathcal{A})]_{n+i,n+i} = [\cosh(\sqrt{A^TA})]_{ii}, \qquad 1 \leq i \leq n,$$

as the authority centrality of node i. The hub communicability between the nodes i and j, $i \neq j$, is defined as

$$[\exp(\mathcal{A})]_{ij} = [\cosh(\sqrt{AA^T})]_{ij}, \qquad 1 \leq i, j \leq n,$$

and the authority communicability between the nodes i and j, $i \neq j$, is given by

$$[\exp(\mathcal{A})]_{n+i,n+j} = [\cosh(\sqrt{A^T A})]_{ij}, \qquad 1 \leq i,j \leq n.$$

Analogously, the hub-authority communicability between the nodes i and j is defined as

$$[\exp(\mathcal{A})]_{i,n+j} = [A\,\mathrm{s}(\sqrt{AA^T})]_{ij} = [\mathrm{s}(\sqrt{AA^T})A^T]_{ji} = \exp(\mathcal{A})_{n+j,i}, \qquad 1 \leq i,j \leq n.$$

This is also the authority-hub communicability between the nodes j and i.

When the graph G has many nodes and, therefore, the adjacency matrix A is large, direct evaluation of $\exp(\mathcal{A})$ generally is not feasible. Benzi et al. [11] discuss how to apply Gauss-type quadrature rules to determine upper and lower bounds for expressions of the form

$$\mathbf{u}^T \exp(\mathcal{A})\mathbf{v}, \qquad \mathbf{u}, \mathbf{v} \in \mathbb{R}^{2n}. \tag{2.4.4}$$

Note that the hub and authority centralities as well as the hub-authority communicability can be expressed in the form (2.4.4) for suitable vectors $\mathbf{u}$ and $\mathbf{v}$. The possibility of determining upper and lower bounds for expressions of the form (2.4.4) by applying a few steps of the symmetric Lanczos method to $\mathcal{A}$ and interpreting the tridiagonal matrix obtained as a Gauss quadrature rule was first observed by Golub [56]. A simple modification of the tridiagonal matrix gives an associated Gauss–Radau rule with a specified quadrature node. Pairs of a Gauss rule and a suitably chosen Gauss–Radau rule provide upper and lower bounds for (2.4.4). A detailed description of this approach and many applications can be found in the nice book by Golub and Meurant [59]; see also [58]. Benzi and Boito [10] were the first to apply this technique to studying undirected graphs.

The application of Gauss-type quadrature rules is attractive when bounds for only a few quantities (2.4.4) are to be computed. However, when bounds for many hub and authority centralities or hub-authority communicabilities are desired, then the evaluation of all the Gauss and Gauss–Radau rules required can be expensive, because the computational work is proportional to the number of bounds desired.

For instance, when we would like to determine one or a few nodes with the largest hub centrality in a large graph, upper and lower bounds for all of the first n diagonal entries of $\exp(\mathcal{A})$ have to be computed in order to be able to ascertain which node(s) have the largest hub centrality. It is even more expensive to determine the node(s) with the largest hub-authority communicability, because this requires the evaluation of bounds for all the entries of $\exp(\mathcal{A})$ above the diagonal with indices $1 \leq i \leq n$ and $n+1 \leq j \leq 2n$, i.e., of $n^2 - 1$ expression of the form (2.4.4).

As in the symmetric case, the computational effort can be reduced using a low-rank approximation of the adjacency matrix A. Consider for the moment the (full) SVD,

$$A = U\Sigma V^T = \sum_{j=1}^{n} \sigma_j \mathbf{u}_j \mathbf{v}_j^T, \tag{2.4.5}$$

where the matrices $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n\times n}$ and $V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n] \in \mathbb{R}^{n\times n}$ are orthogonal and the diagonal matrix

$$\Sigma = \mathrm{diag}[\sigma_1, \sigma_2, \ldots, \sigma_n] \in \mathbb{R}^{n\times n}, \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0,$$

contains the singular values. We refer to the singular triplets $\{\sigma_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^N$ associated with the $N \leq n$ largest singular values of A as the N largest singular triplets. The numerical examples of Section 2.4.2 illustrate that many adjacency matrices can be approximated fairly accurately by a matrix of low rank $N \ll n$ made up of the N largest singular triplets. We use this rank-N approximation to identify a subset of nodes that contains the desired nodes, such as the five nodes with the largest hub centrality, and then compute improved bounds with the aid of Gauss-type quadrature rules for the nodes in the determined subset. This hybrid approach can be much cheaper for large graphs than only using Gauss quadrature. The latter approach, in turn, is much cheaper than evaluating the matrix exponential $\exp(\mathcal{A})$. This is illustrated in Section 2.4.2. The hybrid method generalizes the scheme proposed in Section 2.3.1 for undirected graphs to directed ones.

2.4.1 Use of low-rank approximation

In this section, we derive bounds for expressions of the form

$$\mathbf{z}^T\exp(\mathcal{A})\mathbf{w}, \qquad \mathbf{z} = \begin{bmatrix}\mathbf{z}_1\\ \mathbf{z}_2\end{bmatrix}, \qquad \mathbf{w} = \begin{bmatrix}\mathbf{w}_1\\ \mathbf{w}_2\end{bmatrix}, \qquad \mathbf{z}_1, \mathbf{z}_2, \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^n, \tag{2.4.6}$$

that are computable with a partial SVD of the adjacency matrix A. Using the SVD of A (2.4.5), we obtain

$$\mathcal{A} = \begin{bmatrix}U & 0\\ 0 & V\end{bmatrix}\begin{bmatrix}0 & \Sigma\\ \Sigma & 0\end{bmatrix}\begin{bmatrix}U^T & 0\\ 0 & V^T\end{bmatrix}$$

and

$$\begin{aligned}\exp(\mathcal{A}) &= \begin{bmatrix}U & 0\\ 0 & V\end{bmatrix}\exp\left(\begin{bmatrix}0 & \Sigma\\ \Sigma & 0\end{bmatrix}\right)\begin{bmatrix}U^T & 0\\ 0 & V^T\end{bmatrix}\\ &= \begin{bmatrix}U & 0\\ 0 & V\end{bmatrix}\begin{bmatrix}\cosh(\Sigma) & \sinh(\Sigma)\\ \sinh(\Sigma) & \cosh(\Sigma)\end{bmatrix}\begin{bmatrix}U^T & 0\\ 0 & V^T\end{bmatrix}\\ &= \begin{bmatrix}\sum_{k=1}^n \cosh(\sigma_k)\mathbf{u}_k\mathbf{u}_k^T & \sum_{k=1}^n \sinh(\sigma_k)\mathbf{u}_k\mathbf{v}_k^T\\ \sum_{k=1}^n \sinh(\sigma_k)\mathbf{v}_k\mathbf{u}_k^T & \sum_{k=1}^n \cosh(\sigma_k)\mathbf{v}_k\mathbf{v}_k^T\end{bmatrix}.\end{aligned} \tag{2.4.7}$$

It follows that

$$\mathbf{z}^T\exp(\mathcal{A})\mathbf{w} = \sum_{k=1}^n \cosh(\sigma_k)\left[\bar{z}_k\bar{w}_k + \tilde{z}_k\tilde{w}_k\right] + \sinh(\sigma_k)\left[\bar{z}_k\tilde{w}_k + \tilde{z}_k\bar{w}_k\right], \tag{2.4.8}$$

where

$$\begin{aligned}\bar{z}_k &= \mathbf{z}_1^T\mathbf{u}_k, & \tilde{z}_k &= \mathbf{z}_2^T\mathbf{v}_k,\\ \bar{w}_k &= \mathbf{w}_1^T\mathbf{u}_k, & \tilde{w}_k &= \mathbf{w}_2^T\mathbf{v}_k.\end{aligned} \tag{2.4.9}$$

Using the N largest singular triplets, we can approximate the bilinear form (2.4.8) by

$$F_{\mathbf{z},\mathbf{w}}^{(N)} := \sum_{k=1}^N \cosh(\sigma_k)\left[\bar{z}_k\bar{w}_k + \tilde{z}_k\tilde{w}_k\right] + \sinh(\sigma_k)\left[\bar{z}_k\tilde{w}_k + \tilde{z}_k\bar{w}_k\right]. \tag{2.4.10}$$

Throughout this thesis $\|\cdot\|$ denotes the Euclidean vector norm.

Theorem 2.4.1. *Let $\{\sigma_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^N$ denote the N largest singular triplets of A. Let the vectors $\mathbf{z}$ and $\mathbf{w}$ be split according to (2.4.6). Then we have the bounds*

$$L_{\mathbf{z},\mathbf{w}}^{(N)} \le \mathbf{z}^T\exp(\mathcal{A})\mathbf{w} \le U_{\mathbf{z},\mathbf{w}}^{(N)}, \tag{2.4.11}$$

with

$$\begin{aligned} L^{(N)}_{\mathbf{z},\mathbf{w}} &:= F^{(N)}_{\mathbf{z},\mathbf{w}} - G^{(N)}_{\mathbf{z},\mathbf{w}}, \\ U^{(N)}_{\mathbf{z},\mathbf{w}} &:= F^{(N)}_{\mathbf{z},\mathbf{w}} + G^{(N)}_{\mathbf{z},\mathbf{w}}, \end{aligned} \tag{2.4.12}$$

where $F^{(N)}_{\mathbf{z},\mathbf{w}}$ is defined by (2.4.10) *and*

$$\begin{aligned} G^{(N)}_{\mathbf{z},\mathbf{w}} = \cosh(\sigma_N) &\left[\|\bar{\mathbf{z}}^{(N)}\| \, \|\bar{\mathbf{w}}^{(N)}\| + \|\tilde{\mathbf{z}}^{(N)}\| \, \|\tilde{\mathbf{w}}^{(N)}\| \right] \\ + \sinh(\sigma_N) &\left[\|\bar{\mathbf{z}}^{(N)}\| \, \|\tilde{\mathbf{w}}^{(N)}\| + \|\tilde{\mathbf{z}}^{(N)}\| \, \|\bar{\mathbf{w}}^{(N)}\| \right]. \end{aligned} \tag{2.4.13}$$

Here we use the notation

$$\mathbf{x}^{(N)} = [x_{N+1}, \ldots, x_n]^T$$

for any $\mathbf{x} \in \mathbb{R}^n$. The entries $\bar{z}_k$, $\tilde{z}_k$, $\bar{w}_k$, $\tilde{w}_k$ of the vectors $\bar{\mathbf{z}}^{(N)}$, $\tilde{\mathbf{z}}^{(N)}$, $\bar{\mathbf{w}}^{(N)}$, $\tilde{\mathbf{w}}^{(N)}$ are given by (2.4.9)*.*

Proof. Let $c_k = \cosh(\sigma_k)$ and $s_k = \sinh(\sigma_k)$. The functions $\cosh(x)$ and $\sinh(x)$ are nondecreasing and nonnegative for $x \geq 0$. Therefore the Cauchy and triangle inequalities yield

$$\begin{aligned} \left| \mathbf{z}^T \exp(\mathcal{A}) \mathbf{w} - F^{(N)}_{\mathbf{z},\mathbf{w}} \right| &= \left| \sum_{k=N+1}^{n} c_k \left[\bar{z}_k \bar{w}_k + \tilde{z}_k \tilde{w}_k \right] + s_k \left[\bar{z}_k \tilde{w}_k + \tilde{z}_k \bar{w}_k \right] \right| \\ &\leq c_N \sum_{k=N+1}^{n} |\bar{z}_k \bar{w}_k + \tilde{z}_k \tilde{w}_k| + s_N \sum_{k=N+1}^{n} |\bar{z}_k \tilde{w}_k + \tilde{z}_k \bar{w}_k| \\ \leq c_N \sum_{k=N+1}^{n} \left(|\bar{z}_k| \, |\bar{w}_k| + |\tilde{z}_k| \, |\tilde{w}_k| \right) &+ s_N \sum_{k=N+1}^{n} \left(|\bar{z}_k| \, |\tilde{w}_k| + |\tilde{z}_k| \, |\bar{w}_k| \right) \\ \leq c_N \Bigg[\left(\sum_{k=N+1}^{n} \bar{z}_k^2 \right)^{\frac{1}{2}} \left(\sum_{k=N+1}^{n} \bar{w}_k^2 \right)^{\frac{1}{2}} &+ \left(\sum_{k=N+1}^{n} \tilde{z}_k^2 \right)^{\frac{1}{2}} \left(\sum_{k=N+1}^{n} \tilde{w}_k^2 \right)^{\frac{1}{2}} \Bigg] \\ + s_N \Bigg[\left(\sum_{k=N+1}^{n} \bar{z}_k^2 \right)^{\frac{1}{2}} \left(\sum_{k=N+1}^{n} \tilde{w}_k^2 \right)^{\frac{1}{2}} &+ \left(\sum_{k=N+1}^{n} \tilde{z}_k^2 \right)^{\frac{1}{2}} \left(\sum_{k=N+1}^{n} \bar{w}_k^2 \right)^{\frac{1}{2}} \Bigg], \end{aligned}$$

from which (2.4.11) follows. □

We have formulated the bounds (2.4.11) so that they can be evaluated when the N largest singular triplets of A are known; see below. It is possible to sharpen the bounds by replacing σ_N by σ_{N+1} in (2.4.13), but then knowledge of the N largest singular triplets of A is not sufficient to compute the bounds.

Corollary 2.4.1. *Let the norm of the vectors $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{w}_1$, $\mathbf{w}_2$ and the N largest singular triplets of A be known. Then the bounds (2.4.11) can be evaluated.*

Proof. It follows from (2.4.9) that $\|\bar{\mathbf{z}}\| = \|\mathbf{z}_1\|$ and $\|\tilde{\mathbf{z}}\| = \|\mathbf{z}_2\|$. Therefore,

$$\|\bar{\mathbf{z}}^{(N)}\| = \left(\|\mathbf{z}_1\|^2 - \sum_{k=1}^{N} \bar{z}_k^2 \right)^{\frac{1}{2}}, \qquad \|\tilde{\mathbf{z}}^{(N)}\| = \left(\|\mathbf{z}_2\|^2 - \sum_{k=1}^{N} \tilde{z}_k^2 \right)^{\frac{1}{2}},$$

and similarly for the vectors $\|\bar{\mathbf{w}}^{(N)}\|$ and $\|\tilde{\mathbf{w}}^{(N)}\|$. Substituting these expressions into the right-hand side of (2.4.13) shows the desired result. □

Remark 2.4.1. If either one of the two halves $\mathbf{z}_1$ or $\mathbf{z}_2$ of the vector $\mathbf{z}$ vanish, then the bounds of Theorem 2.4.1 simplify. The same is true for the vector $\mathbf{w}$. For example, if $\mathbf{z}_2 = \mathbf{0}$, then

$$G_{\mathbf{z},\mathbf{w}}^{(N)} = \cosh(\sigma_N)\,\|\bar{\mathbf{z}}^{(N)}\|\,\|\bar{\mathbf{w}}^{(N)}\| + \sinh(\sigma_N)\,\|\bar{\mathbf{z}}^{(N)}\|\,\|\tilde{\mathbf{w}}^{(N)}\|.$$

Corollary 2.4.2. *Assume that the conditions of Theorem 2.4.1 hold. Then*

$$F_{\mathbf{z},\mathbf{z}}^{(N)} \le \mathbf{z}^T \exp(\mathcal{A})\mathbf{z} \le U_{\mathbf{z},\mathbf{z}}^{(N)}. \tag{2.4.14}$$

Proof. For each k every pair of terms in the right-hand side of (2.4.8) is nonnegative. This is a consequence of $\cosh(\sigma) > \sinh(\sigma) \ge 0$ for $\sigma \ge 0$ and $\bar{z}_k^2+\tilde{z}_k^2+2\bar{z}_k\tilde{z}_k = (\bar{z}_k+\tilde{z}_k)^2$. Therefore $F_{\mathbf{z},\mathbf{z}}^{(N)}$ is a lower bound. The upper bound is established by Theorem 2.4.1. □

Corollary 2.4.3. *Let the conditions of Theorem 2.4.1 hold. Then the bounds* (2.4.11) *satisfy*

$$\begin{array}{rcl} L_{\mathbf{z},\mathbf{w}}^{(N)} - F_{\mathbf{z},\mathbf{w}}^{(N)} & \le & L_{\mathbf{z},\mathbf{w}}^{(N+1)} - F_{\mathbf{z},\mathbf{w}}^{(N+1)} \le 0, \\ U_{\mathbf{z},\mathbf{w}}^{(N)} - F_{\mathbf{z},\mathbf{w}}^{(N)} & \ge & U_{\mathbf{z},\mathbf{w}}^{(N+1)} - F_{\mathbf{z},\mathbf{w}}^{(N+1)} \ge 0, \end{array} \tag{2.4.15}$$

for $1 \le N < n$. *Under the assumptions of Corollary 2.4.2, the bounds* (2.4.14) *satisfy*

$$F_{\mathbf{z},\mathbf{w}}^{(N)} \le F_{\mathbf{z},\mathbf{w}}^{(N+1)}, \qquad U_{\mathbf{z},\mathbf{w}}^{(N)} \ge U_{\mathbf{z},\mathbf{w}}^{(N+1)}, \qquad 1 \le N < n. \tag{2.4.16}$$

Proof. The inequality $\|\mathbf{x}^{(N)}\| \ge \|\mathbf{x}^{(N+1)}\|$ yields

$$\begin{aligned} G_{\mathbf{z},\mathbf{w}}^{(N)} &= F_{\mathbf{z},\mathbf{w}}^{(N)} - L_{\mathbf{z},\mathbf{w}}^{(N)} \\ &= \cosh(\sigma_N)\left[\|\bar{\mathbf{z}}^{(N)}\|\,\|\bar{\mathbf{w}}^{(N)}\| + \|\tilde{\mathbf{z}}^{(N)}\|\,\|\tilde{\mathbf{w}}^{(N)}\|\right] \\ &\quad + \sinh(\sigma_N)\left[\|\bar{\mathbf{z}}^{(N)}\|\,\|\tilde{\mathbf{w}}^{(N)}\| + \|\tilde{\mathbf{z}}^{(N)}\|\,\|\bar{\mathbf{w}}^{(N)}\|\right] \\ &\ge \cosh(\sigma_{N+1})\left[\|\bar{\mathbf{z}}^{(N+1)}\|\,\|\bar{\mathbf{w}}^{(N+1)}\| + \|\tilde{\mathbf{z}}^{(N+1)}\|\,\|\tilde{\mathbf{w}}^{(N+1)}\|\right] \\ &\quad + \sinh(\sigma_{N+1})\left[\|\bar{\mathbf{z}}^{(N+1)}\|\,\|\tilde{\mathbf{w}}^{(N+1)}\| + \|\tilde{\mathbf{z}}^{(N+1)}\|\,\|\bar{\mathbf{w}}^{(N+1)}\|\right] = G_{\mathbf{z},\mathbf{w}}^{(N+1)}. \end{aligned}$$

The inequalities (2.4.15) are a consequence of $F_{\mathbf{z},\mathbf{w}}^{(N)} - L_{\mathbf{z},\mathbf{w}}^{(N)} = U_{\mathbf{z},\mathbf{w}}^{(N)} - F_{\mathbf{z},\mathbf{w}}^{(N)}$, and the inequalities (2.4.16) follow from the fact that $\cosh(x)$ and $\sinh(x)$ are nonnegative and nondecreasing functions for $x \ge 0$. □

The above bounds are of particular interest when the vectors $\mathbf{z}$ and $\mathbf{w}$ are axis vectors. We therefore provide expressions for this situation. Also, the case when $\mathbf{w}_1$ has all entries equal is considered. Let $\mathbf{e}_i = [0,\ldots,0,1,0,\ldots,0]^T$ denote the ith axis vector of appropriate dimension. When $\mathbf{z} = \mathbf{e}_i$ and $\mathbf{w} = \mathbf{e}_j$, we can write

$$[\exp(\mathcal{A})]_{ij} = \begin{cases} \sum_{k=1}^n \cosh(\sigma_k) u_{ik} u_{jk}, & 1 \le i,j \le n, \\ \sum_{k=1}^n \sinh(\sigma_k) u_{ik} v_{j-n,k}, & 1 \le i \le n,\ n+1 \le j \le 2n, \\ \sum_{k=1}^n \sinh(\sigma_k) v_{i-n,k} u_{j,k}, & n+1 \le i \le 2n,\ 1 \le j \le n, \\ \sum_{k=1}^n \cosh(\sigma_k) v_{i-n,k} v_{j-n,k}, & n+1 \le i,j \le 2n, \end{cases}$$

where $\mathbf{u}_k = [u_{1k}, u_{2k}, \ldots, u_{nk}]^T$ and $\mathbf{v}_k = [v_{1k}, v_{2k}, \ldots, v_{nk}]^T$ are the kth left and right singular vectors of A, respectively.

Letting $\mathbf{z} = \mathbf{w} = \mathbf{e}_i$, $i = 1, \dots, n$, yields hub centralities, and $\mathbf{z} = \mathbf{w} = \mathbf{e}_i$, $i = n+1, \dots, 2n$, gives authority centralities. We obtain the bounds

$$F_{ii}^{(N)} \leq \left[e^{\mathcal{A}}\right]_{ii} \leq U_{ii}^{(N)}, \qquad 1 \leq i \leq 2n, \tag{2.4.17}$$

where

$$F_{ii}^{(N)} := \begin{cases} \sum_{k=1}^{N} \cosh(\sigma_k) u_{ik}^2, & 1 \leq i \leq n, \\ \sum_{k=1}^{N} \cosh(\sigma_k) v_{i-n,k}^2, & n+1 \leq i \leq 2n, \end{cases}$$

$$U_{ii}^{(N)} := F_{ii}^{(N)} + \begin{cases} \cosh(\sigma_N) \left(\mathcal{U}_i^{(N)}\right)^2, & 1 \leq i \leq n, \\ \cosh(\sigma_N) \left(\mathcal{V}_{i-n}^{(N)}\right)^2, & n+1 \leq i \leq 2n, \end{cases}$$

and

$$\mathcal{U}_r^{(N)} = \left(1 - \sum_{k=1}^{N} u_{rk}^2\right)^{\frac{1}{2}}, \qquad \mathcal{V}_r^{(N)} = \left(1 - \sum_{k=1}^{N} v_{rk}^2\right)^{\frac{1}{2}}.$$

Letting $\mathbf{z} = \mathbf{e}_i$ and $\mathbf{w} = \mathbf{e}_j$, $i, j = 1, \dots, n$, $i \neq j$, determines hub communicabilities, and the choice $\mathbf{z} = \mathbf{e}_i$ and $\mathbf{w} = \mathbf{e}_j$, $i, j = n+1, \dots, 2n$, $i \neq j$, gives authority communicabilities. We have the bounds

$$L_{ij}^{(N)} \leq [\exp(\mathcal{A})]_{ij} \leq U_{ij}^{(N)}, \qquad i \neq j, \tag{2.4.18}$$

where $L_{ij}^{(N)} = F_{ij}^{(N)} - G_{ij}^{(N)}$ and $U_{ij}^{(N)} = F_{ij}^{(N)} + G_{ij}^{(N)}$, with

$$F_{ij}^{(N)} := \begin{cases} \sum_{k=1}^{N} \cosh(\sigma_k) u_{ik} u_{jk}, & 1 \leq i, j \leq n, \\ \sum_{k=1}^{N} \cosh(\sigma_k) v_{i-n,k} v_{j-n,k}, & n+1 \leq i, j \leq 2n, \end{cases}$$

$$G_{ij}^{(N)} := \begin{cases} \cosh(\sigma_N)\, \mathcal{U}_i^{(N)} \mathcal{U}_j^{(N)}, & 1 \leq i, j \leq n, \\ \cosh(\sigma_N)\, \mathcal{V}_{i-n}^{(N)} \mathcal{V}_{j-n}^{(N)}, & n+1 \leq i, j \leq 2n. \end{cases}$$

Setting $\mathbf{z} = \mathbf{e}_i$, $i = 1, \dots, n$, $\mathbf{w}_1 = n^{-\frac{1}{2}}[1, \dots, 1]^T$, and $\mathbf{w}_2 = \mathbf{0}$, we can write

$$n^{-\frac{1}{2}} \sum_{j=1}^{n} [\exp(\mathcal{A})]_{ij} = n^{-\frac{1}{2}} \sum_{k=1}^{n} \cosh(\sigma_k)\, u_{ik} \sum_{j=1}^{n} u_{jk}, \qquad i = 1, \dots, n, \tag{2.4.19}$$

which is a measure of the importance of a node as a hub. This concept is connected to the *total subgraph communicability* for a node of an undirected network, defined in [12], and to the *f-starting convenience* for a node of a directed network, introduced in [50]. We have

$$L_{\mathbf{z},\mathbf{w}}^{(N)} \leq n^{-\frac{1}{2}} \sum_{j=1}^{n} [\exp(\mathcal{A})]_{ij} \leq U_{\mathbf{z},\mathbf{w}}^{(N)}, \tag{2.4.20}$$

where the bounds are given by (2.4.12), with

$$F_{\mathbf{z},\mathbf{w}}^{(N)} := n^{-\frac{1}{2}} \sum_{k=1}^{N} \cosh(\sigma_k)\, u_{ik} \sum_{j=1}^{n} u_{jk},$$

$$G_{\mathbf{z},\mathbf{w}}^{(N)} := n^{-\frac{1}{2}} \cosh(\sigma_N) \mathcal{U}_i^{(N)} \left(n - \sum_{k=1}^{N} \left(\sum_{j=1}^{n} u_{jk}\right)^2\right)^{\frac{1}{2}}.$$

Analogously, setting $\mathbf{z}_1 = \mathbf{0}$, $\mathbf{z}_2 = n^{-\frac{1}{2}}[1,\dots,1]^T$, and $\mathbf{w} = \mathbf{e}_j$, $j = n+1,\dots,2n$, we obtain

$$n^{-\frac{1}{2}} \sum_{i=n+1}^{2n} [\exp(\mathcal{A})]_{ij} = n^{-\frac{1}{2}} \sum_{k=1}^{n} \cosh(\sigma_k) v_{j-n,k} \sum_{i=1}^{n} v_{ik}, \qquad j = n+1,\dots,2n,$$

which is a measure of the importance of a node as an authority; see also the definition of *f-ending convenience* in [50]. We have

$$L^{(N)}_{\mathbf{z},\mathbf{w}} \le n^{-\frac{1}{2}} \sum_{i=n+1}^{2n} [\exp(\mathcal{A})]_{ij} \le U^{(N)}_{\mathbf{z},\mathbf{w}},$$

where the bounds are given by (2.4.12), with

$$F^{(N)}_{\mathbf{z},\mathbf{w}} := n^{-\frac{1}{2}} \sum_{k=1}^{N} \cosh(\sigma_k) v_{j-n,k} \sum_{i=1}^{N} v_{ik},$$

$$G^{(N)}_{\mathbf{z},\mathbf{w}} := n^{-\frac{1}{2}} \cosh(\sigma_N)\, \mathcal{V}^{(N)}_{j-n} \left(n - \sum_{k=1}^{N} \left(\sum_{i=1}^{n} v_{ik} \right)^2 \right)^{\frac{1}{2}}.$$

We conclude this section with two observations. It suffices that the function f be nondecreasing and nonnegative on the spectrum of $\mathcal{A}$ in order to establish upper and lower bounds for $\mathbf{z}^T f(\mathcal{A})\mathbf{w}$ for given vectors $\mathbf{z}$ and $\mathbf{w}$. This is discussed in [51]. The fact that f is the exponential function yields the particular structure exhibited in (2.4.7). Moreover, the situation $\mathbf{z} \ne \mathbf{w}$ also can be dealt with by using the relation

$$\mathbf{z}^T \exp(\mathcal{A})\mathbf{w} = \frac{1}{4}(\mathbf{z}+\mathbf{w})^T \exp(\mathcal{A})(\mathbf{z}+\mathbf{w}) - \frac{1}{4}(\mathbf{z}-\mathbf{w})^T \exp(\mathcal{A})(\mathbf{z}-\mathbf{w}). \qquad (2.4.21)$$

Thus, we can apply Corollary 2.4.2 to determine upper and lower bounds for the left-hand side expression. Computed examples of Section 2.4.2 illustrate the performance of the bounds of this section.

Determining important nodes by partial singular value decomposition. Now we describe how knowledge of the N leading singular triplets $\{\sigma_k, \mathbf{u}_k, \mathbf{v}_k\}_{k=1}^N$ of A and the bounds (2.4.17) can be used to determine two subsets of nodes that contain the nodes with the largest hub centrality $[\cosh(\sqrt{AA^T})]_{ii}$ and the nodes with the largest authority centrality $[\cosh(\sqrt{A^T A})]_{ii}$, respectively. The approach is analogous to the one applied in Section 2.3 to determine a low-rank approximant of a large symmetric adjacency matrix. However, several aspects of the method are different due to the fact that the adjacency matrices considered in this Section are not symmetric.

Let $F^{(N)}_{ii}$ and $U^{(N)}_{ii}$ be the lower and upper bounds (2.4.17), respectively, and let $\mathcal{L}^{(N)}_{H,m}$ denote the mth largest lower bound $F^{(N)}_{ii}$ for $1 \le i \le n$. Introduce the index sets

$$S^{(N)}_{H,m} = \left\{ i \;:\; 1 \le i \le n \text{ and } U^{(N)}_{ii} \ge \mathcal{L}^{(N)}_{H,m} \right\}, \qquad N = 1,2,\dots,n. \qquad (2.4.22)$$

This set is of interest when ranking nodes according to their hub centrality.

Let $\mathcal{L}^{(N)}_{A,m}$ denote the mth largest lower bound $F^{(N)}_{ii}$ for $n+1 \le i \le 2n$ and consider the index sets

$$S^{(N)}_{A,m} = \left\{ i \;:\; n+1 \le i \le 2n \text{ and } U^{(N)}_{ii} \ge \mathcal{L}^{(N)}_{A,m} \right\}, \qquad N = 1,2,\dots,n. \qquad (2.4.23)$$

We use this set to determine nodes with the largest authority centrality.

The computations required to determine the most important hubs and the most important authorities are similar. We therefore can treat both computations simultaneously and omit the subscripts H and A for the sets (2.4.22) and (2.4.23). It is not hard to see that if $i \notin S_m^{(N)}$, then node i cannot be in the subset of the m nodes with the largest centrality. Moreover, any node whose index is an element of $S_m^{(N)}$ can be in this subset. Let $|S_m^{(N)}|$ denote the cardinality of $S_m^{(N)}$. The following inequalities are useful in our computations.

Corollary 2.4.4.

$$S_m^{(n)} \subseteq S_m^{(n-1)} \subseteq \cdots \subseteq S_m^{(1)} \qquad \text{and} \qquad |S_m^{(n)}| \geq m. \tag{2.4.24}$$

The set $S_m^{(N)}$ contains the indices for a subset of nodes that contains the set of the m most important nodes. In particular, when $|S_m^{(N)}| = m$, the set $S_m^{(N)}$ contains the indices for the m most important nodes.

Proof. By the definition of the sets $S_m^{(N)}$, each set contains at least m indices. The relations (2.4.24) now follow from (2.4.16) and (2.4.17). The observation about the situation when $|S_m^{(N)}| = m$ is a consequence of the fact that the set $S_m^{(N)}$ contains the indices for the m nodes with the largest centrality. □

Remark 2.4.2. The lower bound $F_{ii}^{(N)}$ of (2.4.17) usually converges to $[\exp(\mathcal{A})]_{ii}$ much faster than the upper bound $U_{ii}^{(N)}$ as N increases. Therefore, for N fixed, $F_{ii}^{(N)}$ typically is a better approximation of $[\exp(\mathcal{A})]_{ii}$ than $\frac{1}{2}(F_{ii}^{(N)} + U_{ii}^{(N)})$.

Remark 2.4.3. The ordering of the lower bounds $F_{ii}^{(N)}$, $i \in S_m^{(N)}$, may be different from the ordering of the centralities $[\exp(\mathcal{A})]_{ii}$.

We turn to some computational issues. Evaluation of the bounds (2.4.17) and (2.4.18) requires the computation of $f(\sigma_N)$. This may result in overflow when the graph contains many nodes and f is the hyperbolic sine or cosine function. For instance, when the computations are carried out in double precision arithmetic, i.e., with about 16 significant decimal digits, we obtain overflow when evaluating $\cosh(x)$ for $x \gtrsim 710$. This difficulty can be circumvented by replacing $\cosh(x)$ by

$$\exp(-\mu)\cosh(x) = \frac{1}{2}(\exp(x-\mu) + \exp(-x-\mu)),$$

for an appropriate value of μ. Since the largest singular triplet $\{\sigma_1, \mathbf{u}_1, \mathbf{v}_1\}$ of A is available, we use $\mu = \sigma_1$ in the computed examples reported in Section 2.4.2.

Another computational difficulty to overcome is that we do not know in advance how many of the largest singular triplets $\{\sigma_k, \mathbf{u}_k, \mathbf{v}_k\}_{k=1}^N$ of A have to be computed to obtain useful bounds (2.4.17) or (2.4.18). Assume for definiteness that we would like to determine the m nodes with the largest hub centrality. We use the augmented implicitly restarted Golub–Kahan bidiagonalization method described in [7] to compute the largest triplets in the examples of Section 2.4.2. Our implementation of this method is a slight modification of the MATLAB function irlba described in [7]. It differs in that it can be restarted with singular triplets produced by previous calls to the function as input. This modification, which we also refer to as irlba, makes it possible to compute new singular triplets without recomputing already known triplets. In the examples of

Section 2.4.2, we determine the $q = 5$ largest singular triplets that have not yet been computed by repeated calls of irlba. Thus, the first call of irlba yields the singular triplets $\{\sigma_k, \mathbf{u}_k, \mathbf{v}_k\}_{k=1}^{q}$. If $|S_m^{(q)}| > m$, then we call irlba again to determine the next q singular triplets $\{\sigma_k, \mathbf{u}_k, \mathbf{v}_k\}_{k=q+1}^{2q}$. If $|S_m^{(N)}| = m$ for some $N \leq 2q$, then we are done; otherwise we compute the next q singular triplets $\{\sigma_k, \mathbf{u}_k, \mathbf{v}_k\}_{k=2q+1}^{3q}$ of A with irlba, and so on. This approach to computing the N largest singular triplets of A only requires storage of the already computed triplets and a residual vector when the next batch of q singular triplets is to be computed.

The above method for determining singular triplets is attractive when the computer at hand allows storage of all already computed singular triplets. However, when the adjacency matrix is very large and fairly many singular triplets $\{\sigma_k, \mathbf{u}_k, \mathbf{v}_k\}_{k=1}^{N}$ are required for $|S_m^{(N)}| = m$ to hold, the use of a method that requires less computer storage may be preferable. We now outline such a method. Note that the bounds (2.4.17) and (2.4.18) can be updated when a new batch of q singular triplets has been computed. Therefore, the evaluation of the bounds (2.4.17) and (2.4.18) does not require simultaneous access to all computed singular triplets. Hence, we may reduce the storage demand by using an augmented implicitly restarted Golub–Kahan bidiagonalization method that does not require access to all already computed singular triplets to determine the next batch of q singular triplets. The algorithm irblb described in [6] has this property. It uses Leja shifts to determine an acceleration polynomial that dampens singular values outside a region of interest. When calling irblb, the smallest computed singular value σ_N is passed as a parameter. The algorithm then seeks to determine the q singular values that are closest to σ_N and the corresponding singular vectors. This method requires fairly little temporary storage. The memory requirement therefore is much smaller than for the irlba-based computations described above when a large number, N, of singular triplets is needed to secure that $|S_m^{(N)}| = m$. We remark that the use of irblb may require more matrix-vector product evaluations with A and A^T than irlba, because a few of the already computed singular triplets might be recomputed at a subsequent call of irblb. Our computational experience indicates that for many real-world networks the required size of N is quite small and, therefore, irlba generally can be used also for large networks.

In the above discussion, we determined more and more singular triplets of A until their number N is such that

$$|S_m^{(N)}| = m. \tag{2.4.25}$$

We refer to this stopping rule as the *strong convergence criterion*. By Corollary 2.4.4, the set $S_m^{(N)}$ contains the indices of the m nodes with the largest centrality.

The criterion (2.4.25) for choosing N is useful if the required value of N is not too large. We introduce the *weak convergence criterion* to be used for problems for which the large size of N required to satisfy (2.4.25) makes it impractical to compute the associated bounds (2.4.17). The weak convergence criterion is well suited for use with the hybrid algorithm described in Section 2.4.1 for determining the most important nodes. This criterion is designed to stop increasing N when the lower bounds $F_{ii}^{(N)}$ do not increase significantly with N. Specifically, in the case of hub centrality, we stop increasing N when the average increment of the lower bounds $F_{ii}^{(N)}$, $1 \leq i \leq n$, is small when including the Nth singular triplet $\{\sigma_N, \mathbf{u}_N, \mathbf{v}_N\}$ in the bounds. The average contribution of this singular triplet to the bounds $F_{ii}^{(N)}$, $1 \leq i \leq n$, is

$$\frac{1}{n}\sum_{i=1}^{n} f(\sigma_N)u_{i,N}^2 = \frac{1}{n}f(\sigma_N),$$

and we stop increasing N when

$$\frac{1}{n}f(\sigma_N) \leq \tau \cdot \mathcal{L}_{H,m}^{(N)} \tag{2.4.26}$$

for a user-specified tolerance τ. We use $\tau = 10^{-3}$ in the computed examples. Note that when this criterion is satisfied, but not (2.4.25), the m nodes with indices in $S_m^{(N)} = S_{H,m}^{(N)}$ with the largest lower bounds $F_{ii}^{(N)}$ are not guaranteed to be the nodes with the largest hub centrality.

The weak convergence criterion (2.4.26) might yield a set $S_{H,m}^{(N)}$ with many more indices than m, and we may not want to compute accurate bounds for the hub centrality using the approach of Section 1.5 for all nodes with index in $S_{H,m}^{(N)}$. We therefore describe how to determine a smaller index set $\mathcal{J}$, which is likely to contain the indices of the m nodes with the largest hub centrality. In view of that $F_{ii}^{(N)}$ generally is a better approximation of $[f(A)]_{ii}$ than $U_{ii}^{(N)}$ (cf. Remark 2.4.2), we discard from the set $S_{H,m}^{(N)}$ indices for which $F_{ii}^{(N)}$ is much smaller than $\mathcal{L}_{H,m}^{(N)}$. Thus, for a user-chosen parameter $\rho > 0$, we include in the set $\mathcal{J}$ all indices $i \in S_{H,m}^{(N)}$ such that

$$\mathcal{L}_{H,m}^{(N)} - F_{ii}^{(N)} < \rho \cdot \mathcal{L}_{H,m}^{(N)}. \tag{2.4.27}$$

In the computed examples, we use $\rho = 10^{-1}$. We may proceed similarly to prune the set $S_{A,m}^{(N)}$.

Algorithms 3 and 4 describe the computation of the index sets $S_{H,m}^{(N)}$ and $S_{A,m}^{(N)}$, as well as the determination of a suitable value of N. The singular triplets are computed either with the slightly modified MATLAB function irlba from [7] or with algorithm irblb from [6]. The function f is the exponential function (2.4.2); it may be replaced by some other nonnegative nondecreasing function.

The algorithm requires functions for the evaluation of matrix-vector products with the adjacency matrix A and its transpose, its order n, and the desired number of nodes m with the largest hub centrality and authority centrality. In addition, the following parameters have to be provided:

- $N_{\max}$, maximum number of iterations performed;
- q, number of singular triplets computed at each irlba call;
- $M_{\max}$, maximum number of singular vectors kept in memory;
- τ, tolerance used to detect *weak* convergence;
- ρ, tolerance used to construct an extended list of nodes in case of weak convergence; cf. (2.4.27).

Algorithm 3 first initializes the vectors $\boldsymbol{\ell}^H$, $\mathbf{z}^H$, and $\mathbf{s}^H$, whose components at each iteration N are given by

$$\ell_i^H = F_{ii}^{(N)}, \quad z_i^H = U_{ii}^{(N)}, \quad s_i^H = \sum_{k=1}^{N} u_{ik}^2, \quad i = 1, \ldots, n.$$

Algorithm 3 Low-rank approximation, part 1

```
Input: matrix $A$ of size $n$, number $m$ of nodes to be identified,
        tuning constants: $N_{\max}$, $q$, $M_{\max}$, $\tau$, $\rho$
for $i = 1$ to $n$ do $\ell_i^H, \ell_i^A, s_i^H, s_i^A = 0$ end
call irlba to compute $\{\sigma_i, \mathbf{u}_i, \mathbf{v}_i\}_{i=1}^q$ such that $\sigma_i \geq \sigma_{i+1}$
if shift is active then $\mu = \sigma_1$ else $\mu = 0$ end
$N = 0$, $\overline{N} = 0$, flag = true , flag$^H$ = true , flag$^A$ = true
while flag and ($N < \min\{N_{\max}, n\}$) and ($\overline{N} < q$)
    $N = N + 1$, $\overline{N} = \overline{N} + 1$
    $f_\sigma = (\exp(\sigma_N - \mu) + \exp(-\sigma_N - \mu))/2$
    if flag$^H$
        for $i = 1$ to $n$ do $t_i = u_{iN}^2$ end
        $\mathbf{s}^H = \mathbf{s}^H + \mathbf{t}$
        $\boldsymbol{\ell}^H = \boldsymbol{\ell}^H + f_\sigma \cdot \mathbf{t}$
        $\mathbf{z}^H = \boldsymbol{\ell}^H + f_\sigma(1 - \mathbf{s}^H)$
        let $\boldsymbol{\psi} = [\psi_1, \ldots, \psi_n]$ be an index permutation such that $\ell_{\psi_i}^H \geq \ell_{\psi_{i+1}}^H$
        $L_{\max}^H = \ell_{\psi_m}^H$
        $S_{H,m}^{(N)} = \{i \,:\, z_i^H \geq L_{\max}^H\}$
        flag$^H$ = ($|S_{H,m}^{(N)}| > m$) and ($\frac{1}{n} f_\sigma > \tau \cdot L_{\max}^H$)
    end if
    if flag$^A$
        for $i = 1$ to $n$ do $w_i = v_{iN}^2$ end
        $\mathbf{s}^A = \mathbf{s}^A + \mathbf{w}$
        $\boldsymbol{\ell}^A = \boldsymbol{\ell}^A + f_\sigma \cdot \mathbf{w}$
        $\mathbf{z}^A = \boldsymbol{\ell}^A + f_\sigma(1 - \mathbf{s}^A)$
        let $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_n]$ be an index permutation such that $\ell_{\phi_i}^A \geq \ell_{\phi_{i+1}}^A$
        $L_{\max}^A = \ell_{\phi_m}^A$
        $S_{A,m}^{(N)} = \{i \,:\, z_i^A \geq L_{\max}^A\}$
        flag$^A$ = ($|S_{A,m}^{(N)}| > m$) and ($\frac{1}{n} f_\sigma > \tau \cdot L_{\max}^A$)
    end if
    flag = flag$^H$ or flag$^A$
    if flag and $\overline{N} = q$
        if $N < M_{\max}$
            call irlba to compute $\{\sigma_k, \mathbf{u}_k, \mathbf{v}_k\}_{i=N+1}^{N+q}$ such that $\sigma_{N+i} \geq \sigma_{N+i+1}$
            $\overline{N} = 0$
        else
            call irblb to compute sing. values $\nu_1 \geq \cdots \geq \nu_q$ closest to $\sigma_N$
            $r = \arg\min_i |\nu_i - \sigma_N|$
            $\sigma_{N+i} = \nu_{r+i}$, $i = 1, \ldots, q - r$; $\{\mathbf{u}_{N+i}, \mathbf{v}_{N+i}\}$ associated sing. vectors
            $\overline{N} = r$
        end if
    end if
end while
```

The vectors $\boldsymbol{\ell}^A$, $\mathbf{z}^A$, and $\mathbf{s}^A$ for authority centrality, are initialized similarly. Then, irlba is called to compute the first batch of q singular triplets and the main loop is entered. The Boolean variable "flag" is used to signal whether the strong or weak convergence criterion is satisfied. The parameter $N_{\max}$ specifies the maximum number of iterations, i.e., the maximum number of times the loop comprised of lines 7–42 is executed. We found it beneficial to introduce the auxiliary parameter $\overline{N}$ to keep track of how many singular triplets from the current batch are being used. When $\overline{N} = q$, a new batch of singular triplets is computed.

The bounds (2.4.17) are computed in lines 12–14 and 22–24. The vector of indices $\boldsymbol{\psi}$ contains a permutation which yields the lower bounds ℓ_i^H in decreasing order. The set $S_{H,m}^{(N)}$ is constructed at line 17, while the set $S_{A,m}^{(N)}$ is constructed in lines 20–29. Subsequently the exit condition is checked. Lines 31–41 give a new batch of singular triplets. Either one of the two kinds of restarts discussed above may be applied. If $\overline{N}$ is smaller than $M_{\max}$, then we compute the next q singular triplets. If, instead, $\overline{N} \geq M_{\max}$, then we seek to determine the q singular values close to the smallest available singular value σ_N, and we select those singular triplets that have singular values smaller than σ_N; see lines 37–38.

Algorithm 4 Low-rank approximation, part 2

```
43: if flag$^H$
44:     $j = |S_{H,m}^{(N)}| - m$
45:     info = 2    % no convergence
46: else
47:     if $|S_{H,m}^{(N)}| > m$
48:         $\mathcal{J} = \{i \,:\, i > m,\ L_{\max}^H - \ell_{\psi_i}^H < \rho \cdot L_{\max}^H\}$
49:         $j = \min(|\mathcal{J}|, 100)$, $j = \max(j, 5)$
50:         info = 1    % weak convergence
51:     else
52:         $j = 0$
53:         info = 0    % strong convergence
54:     end if
55: end if
56: for $i = 1$ to $m + j$
57:     $\mathcal{N}_i^H = \psi_i$
58:     $\mathcal{V}_i^H = \mathrm{e}^{\mu} \cdot \ell_{\psi_i}^H$
59: end if
60: Output: list of nodes $\mathcal{N}^H$, hub centralities $\mathcal{V}^H$,
61:         spectrum shift $\mu$, iterations $N$, info
```

Algorithm 4 describes the continued computations when the m nodes with the largest hub centrality are desired. The m nodes with the largest authority centrality can be computed in a similar way. Lines 43–55 of the algorithm determine whether the strong or weak convergence criterion is satisfied. The variable "info" contains this information. A list of the desired nodes $\mathcal{N}^H$ is formed in lines 56–59, where the hub centralities also are updated, keeping in mind the spectrum shift at line 9 of Algorithm 3. We remark that the MATLAB implementation of Algorithms 3 and 4 contains some features not described here. For instance, we only apply the correction due to the spectrum shift when this does not cause overflow.

This section described how to determine the m nodes with the largest hub centrality and authority centrality of a large network by using the bounds (2.4.11) and (2.4.14) with $\mathbf{w} = \mathbf{z}$. These bounds can be used in a similar way with $\mathbf{w} \neq \mathbf{z}$ to compute upper and lower bounds for hub and authority communicability. This allows us to determine subsets of nodes with the largest hub or authority communicability; see [11, 50] for other approaches to determine hub and authority communicability.

The hybrid method The computations with the hybrid method already have been commented on. The method first evaluates a partial singular value decomposition of the adjacency matrix A and applies it to determine which nodes might be the most interesting ones with respect to the criterion chosen. This is described above for the situation when we would like to determine the nodes of a large network with the largest hub and authority centrality. More accurate upper and lower bounds for expressions of the form (2.4.6) for the nodes singled out are then determined with the aid of Gauss quadrature rules. This allows us to compute the node with the largest hub and authority centrality of a large graph without evaluating pairs of Gauss and Gauss–Radau rules for every node of the network, i.e., for every diagonal entry of the matrix (2.4.1). The evaluation of Gauss-type rules for every diagonal entry can be expensive for large graphs. The computations with the hybrid method typically are considerably cheaper. This is illustrated in the following section.

2.4.2 Numerical experiments

This section presents a few examples that illustrate the performance of the methods discussed in the previous Section. All computations were carried out in MATLAB version 8.1 (R2013a) 64-bit for Linux, in double precision arithmetic, on an Intel Core i7-860 computer, with 8 Gb RAM. The function f is the exponential function (2.4.2). We applied the methods previously discussed to eight directed unweighted networks coming from the following real-world applications:

Airlines (235 nodes, 2101 edges) represents air traffic and is available at [55]. The nodes represent airports and the directed edges represent flights between them.

Celegans (306 nodes, 2345 edges) is the metabolic network of *Caenorhabditis elegans* [38], a small nematode (roundworm). The data set is available at [1].

Air500 (500 nodes, 24009 edges) is a network of worldwide flight connections between the top 500 airports based on total passenger volume [14] during the time from July 1, 2007, to June 30, 2008 [97].

Twitter (3556 nodes, 188712 edges) is part of the Twitter network [55]. The nodes are users and the directed edges are mentions and re-tweets between users.

Wikivote (8297 nodes, 103689 edges) is the network of administrator elections and vote history data: a directed edge from node i to node j indicates that user i voted for user j [90, 91]. The data set is available at SNAP (Stanford Network Analysis Platform) Network Data Sets [120].

PGP (10680 nodes, 24316 edges) represents the giant component of the network of users of the Pretty-Good-Privacy algorithm for secure information interchange [18]. The data set is available at [1].

Wikipedia (49728 nodes, 941425 edges) represents the Italian Wikipedia. In this graph the nodes are articles and the links represent references to other articles. This data set can be downloaded from [93].

Slashdot (82168 nodes, 948464 edges) represents the Slashdot social network (February 2009). A directed edge from node i to node j indicates that user i tagged user j as a friend or a foe [92]. The data set is available at SNAP (Stanford Network Analysis Platform) Network Data Sets [120].

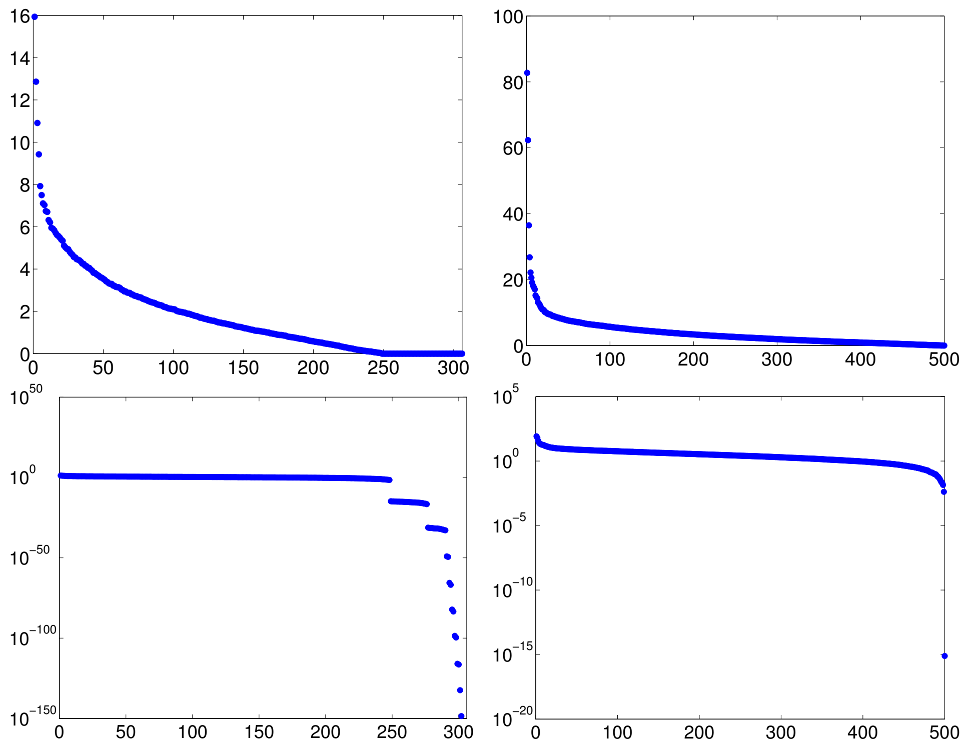


Figure 2.11: Singular values of the Celegans (left) and Air500 (right) test matrices. The graphs in the top row are in decimal scale, while the ones at the bottom are plotted using a semilogarithmic scale.

Figure 2.11 displays the singular values of the Celegans and Air500 test matrices, both in decimal and semilogarithmic scales. These matrices are small enough to allow the computation of all singular values by the MATLAB function svd. The graphs are typical for many adjacency matrices that arise in real applications in the sense that they are numerically rank deficient and all but a fairly small number of singular values can be ignored when evaluating the exponential of the adjacency matrix. The leading 100 singular values of the three largest complex networks considered in the experiments are plotted in Figure 2.12. The singular values are computed by the irlba routine from [7].

Table 2.12 shows for each of the above networks the results obtained by the low-rank approximation, either with the strong or the weak convergence criterion (labeled LR strong and LR weak, respectively), when identifying the 5 most important hubs and

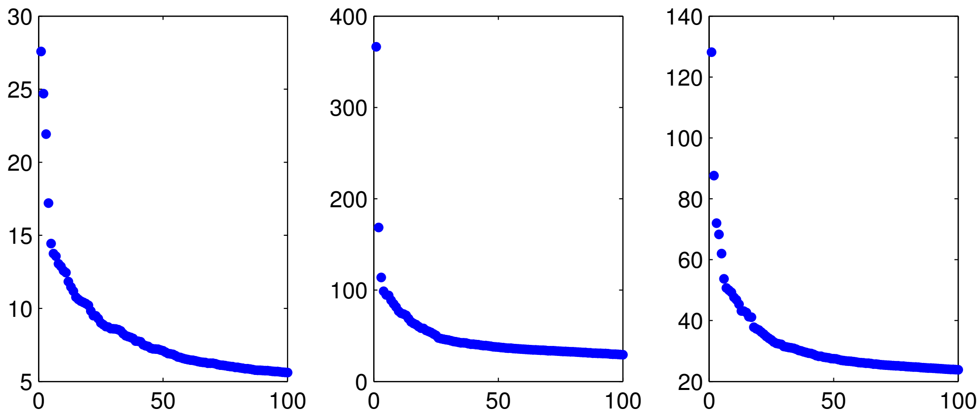


Figure 2.12: Graph of the first 100 singular values of the adjacency matrices of the three largest networks considered in the experiments. From left to right: PGP, Wikipedia, and Slashdot.

the 5 most important authorities in the order of importance. At each iteration we have two sets of candidate nodes, namely $S_{H,5}^{(N)}$ and $S_{A,5}^{(N)}$, which contain the indices of the nodes with the largest hub centrality and the indices of the nodes with the largest authority centrality, respectively. We terminate the computations when $|S_{H,5}^{(N)}| = 5$ for the hub nodes and when $|S_{A,5}^{(N)}| = 5$ for the authority nodes. By Corollary 2.4.4, the set $S_{H,5}^{(N)}$ ($S_{A,5}^{(N)}$) contains the indices of the 5 nodes with the largest hub centrality (authority centrality). The entry of the column labeled "fail" is set to "1" when the relative sizes of the lower bounds $F_{ii}^{(N)}$ (2.4.17) are not in agreement with the exact relative sizes of the hub centralities of the nodes; the entry is "0" when ordering of the lower bounds agrees with the ordering of the exact hub centralities. The exact values of the hub centralities are computed with the MATLAB function expm when $n \le 5 \cdot 10^3$ and by Gauss quadrature rules when $n > 5 \cdot 10^3$. Column 4 of Table 2.12 shows the number N of singular triplets required to satisfy the strong convergence criterion. The convergence tolerance for the irlba and irblb routines is set to $1 \cdot 10^{-3}$ for all computations of this section.

Columns 5–6 of Table 2.12 are obtained when terminating the low-rank approximation method with the weak convergence criterion (2.4.26). The columns display the number of singular triplets needed to reach convergence and the number of candidate nodes included in the resulting index set $S_{H,5}^{(N)}$. The table shows that for hub nodes, the strong convergence criterion requires the computation of at most the 5 largest singular triplets. The weak convergence criterion results in index sets with at most 10 nodes. The results for authority nodes, reported in columns 7–10, are very similar. The table shows that no more than 4 triplets are needed to satisfy the strong convergence criterion and the maximum number of candidate nodes obtained with the weak convergence criterion is 15.

Table 2.13 reports the number of matrix-vector product evaluations (mvp) required when determining the 5 nodes with the largest hub centrality and the 5 nodes with the largest authority centrality. Column 3 shows the number of mvps needed for the evaluation of pairs of Gauss and Gauss–Radau quadrature rules as described in Section 1.5. The number of mvps required for computing low-rank approximations

Table 2.12: Results obtained by the low-rank approximation algorithm, with both strong and weak convergence criteria, when determining the 5 most important hubs and authorities. The table reports the number of failures, the number N of triplets required to reach convergence, and, in case of weak convergence, the cardinality of the lists $S_{H,5}^{(N)}$ and $S_{A,5}^{(N)}$ of candidate nodes.

		hubs				authorities			
		LR strong		LR weak		LR strong		LR weak	
matrix	nodes	fail	N	N	$\lvert S_{H,5}^{(N)}\rvert$	fail	N	N	$\lvert S_{A,5}^{(N)}\rvert$
Airlines	235	0	2	2	5	0	2	2	5
Celegans	306	0	5	3	10	0	3	3	5
Air500	500	0	2	2	5	0	2	2	5
Twitter	3656	0	2	2	5	0	2	2	5
Wikivote	8297	0	2	2	5	0	2	2	5
PGP	10680	0	4	2	10	0	4	2	10
Wikipedia	49728	0	2	2	5	0	2	1	15
Slashdot	82168	0	2	2	5	0	2	2	5

using the strong and weak convergence criteria, as described in Section 2.4.1, are displayed in columns 4–5, and the total number of mvps demanded by the hybrid method of Section 2.4.1 is shown in column 7. All methods compared in Table 2.13 correctly identify the 5 most important nodes of each network. The hybrid method can be seen to require fewer matrix-vector product evaluation than the use of Gauss quadrature rules only.

Table 2.13: Comparison of Gauss quadrature, low rank approximations and the hybrid algorithm. For each adjacency matrix, we report the number of matrix-vector product evaluations required (mvp).

		Gauss	LR strong	LR weak	hybrid	
matrix	nodes	mvp	mvp	mvp	fail	mvp
Airlines	235	1895	30	44	0	84
Celegans	306	2722	44	44	0	104
Air500	500	4914	30	30	0	73
Twitter	3656	38105	30	30	0	84
Wikivote	8297	52871	44	30	0	74
PGP	10680	69012	44	44	0	143
Wikipedia	49728	471917	44	44	0	110
Slashdot	82168	960760	30	44	0	80

Table 2.14 compares the execution times for the approaches of Table 2.13 with the evaluation of the matrix exponential by the MATLAB function expm, which is based on Padé approximation. The table shows the Gauss quadrature approach to be faster than expm, which is too slow to be practical to use for matrices of size larger than 5000×5000. The Gauss quadrature approach is slower than the methods that compute low-rank approximation with a partial singular value decomposition. Moreover, Gauss quadrature and the methods that use low-rank approximations require far less storage space than the expm function, which needs to allocate storage for up to six matrices of

Table 2.14: Comparison of expm, Gauss quadrature, low rank approximations and the hybrid algorithm. For each adjacency matrix, we report the execution time in seconds.

matrix	nodes	expm	Gauss	LR strong	LR weak	hybrid
Airlines	235	5.0e-01	1.0	7.5e-02	6.1e-03	6.1e-02
Celegans	306	5.8e-01	1.3	6.2e-03	5.9e-03	3.4e-02
Air500	500	1.3	5.2	1.1e-01	8.5e-03	1.0e-01
Twitter	3656	1.2e+03	5.9e+01	2.2e-02	2.1e-02	1.0e-01
Wikivote	8297	–	8.0e+01	4.3e-02	2.8e-02	9.1e-02
PGP	10680	–	7.9e+01	5.5e-02	4.2e-02	1.5e-01
Wikipedia	49728	–	5.0e+03	3.1e-01	2.9e-01	1.2
Slashdot	82168	–	1.1e+04	2.7e-01	4.0e-01	8.8e-01

the same size as the input matrix.

We note that even though low-rank approximation with the strong convergence criterion successfully identified the most important nodes in all of our experiments, the ordering of the lower bounds for nodes with indices in the sets $S_{H,m}^{(N)}$ and $S_{A,m}^{(N)}$ is not guaranteed to agree with the ordering of the exact values of the hub or authority centralities; cf. Remark 2.4.3. To secure that the correct order is determined, we apply Gauss quadrature to refine the bounds in the hybrid method. Table 2.14 shows that the computation time is not much larger for the hybrid method than for just computing low-rank approximations.

Table 2.15: Results obtained by the low-rank approximation algorithm, with both strong and weak convergence criteria, when determining the m most important hubs and authorities with m = 1%, 5%, and 10% of the number of nodes in the network. The table reports the number of failures, the number N of singular triplets required to reach convergence, and, in case of weak convergence, the cardinality of the lists $S_{H,m}^{(N)}$ and $S_{A,m}^{(N)}$ of candidate nodes.

		hubs				authorities			
		LR strong		LR weak		LR strong		LR weak	
matrix	m	fail	N	N	$\lvert S_{H,m}^{(N)}\rvert$	fail	N	N	$\lvert S_{A,m}^{(N)}\rvert$
Wikivote	83	0	2	2	83	0	2	2	83
	415	0	2	2	415	0	2	2	415
	830	0	2	2	830	0	2	2	830
PGP	107	0	5	3	112	0	4	3	115
	534	2	26	4	553	8	17	5	542
	1068	124	143	7	1084	248	143	9	1087
Wikipedia	497	0	2	2	497	0	2	2	497
	2486	0	2	2	2486	0	2	2	2486
	4973	0	2	2	4973	0	2	2	4973
Slashdot	822	0	2	2	822	0	2	2	822
	4108	0	2	2	4108	0	2	2	4108
	8217	0	2	2	8217	0	2	2	8217

Table 2.15 investigates the performance of our low-rank approximation methods when more than 5 hubs or authorities are to be identified. Specifically, we seek to

identify the m most important hubs and authorities of the four largest networks in our set of test problems and choose m to be 1%, 5%, and 10% of the number of nodes in the network. It is interesting to note that in most cases the number N of singular triplets necessary to meet the strong and week convergence criteria does not increase with m. In fact, the number of terms in (2.4.10) required to identify a set of nodes does not depend on the number of nodes in the group, but on the topology of the network. The PGP network is the only one for which the strong convergence criterion fails when m is large. The reason for the failure is that the maximum number of terms allowed in (2.4.10) by our code is exceeded. This may depend on many nodes having close values of the hub or authority centralities. However, note that application of the weak convergence criterion produces useful results. The hybrid algorithm based on the weak convergence criterion yields the correct ordering of the m nodes with the largest hub centrality and of the m nodes with the largest authority centrality for all values of m. This experiment indicates that the computing time required to construct the lists $S_{H,m}^{(N)}$ and $S_{A,m}^{(N)}$ does not vary much with m. Of course, the execution time for the refinement phase, in which Gauss quadrature is applied to each node in the lists $S_{H,m}^{(N)}$ and $S_{A,m}^{(N)}$ to improve the bounds, grows linearly with m.

Table 2.16: Computation of starting conveniences by Algorithm 3-4 with the aid of the strong convergence criterion using the bounds (2.4.20) or (2.4.21). We report the number of singular triplets required to satisfy the strong convergence criterion and the number of matrix-vector product evaluations.

		bounds (2.4.20)		bounds (2.4.21)	
matrix	nodes	N	mvp	N	mvp
Airlines	235	2	44	2	44
Celegans	306	5	44	6	112
Air500	500	2	30	2	30
Twitter	3656	2	44	2	44
Wikivote	8297	2	30	2	30
PGP	10680	4	44	5	44
Wikipedia	49728	2	44	2	44
Slashdot	82168	2	30	2	30

Our next example compares the performance of the bounds (2.4.20) and (2.4.21) in the case when $\mathbf{z} \neq \mathbf{w}$. Specifically, we considered the problem of ranking the hubs of a network according to their starting convenience; see (2.4.19). Table 2.16 is determined by applying Algorithm 3-4 in conjunction with the bounds (2.4.20) or with bounds derived from (2.4.21) to identify the 5 most important hubs in a network according to starting convenience. For each network we report the number N of singular triplets required to satisfy the strong convergence criterion and the number of matrix-vector product evaluations. It turns out that the use of the bounds (2.4.20) and those derived from (2.4.21) is essentially equivalent, provided that the evaluation of the two expressions in the right-hand side of (2.4.21) is implemented efficiently, that is, by bounding both expressions simultaneously by using the computed singular triplets. The alternative approach of applying the symmetric Lanczos process to bound each one of the two expressions in the right-hand side of (2.4.21) separately requires more matrix-vector product evaluations with A and A^T. For two of the networks, the bounds obtained from (2.4.21) are less tight and therefore require a larger number of matrix-vector product evaluations. Since we compute the singular triplets in batches of 5, this is only noticeable for the Celegans network in Table 2.16. We illustrate the tightness of the bounds for this network in Figure 2.13. The left-hand side displays the differences between the upper and lower bounds (2.4.20) for all the 306 nodes of the network for the first 5 steps of the algorithm. The top graph on the left-hand side of the figure shows the differences of the upper and lower bounds (2.4.20) for each one of the nodes after one step of the algorithm, the next graph depicts the corresponding differences after two steps, and so on. The bottom graph shows the differences of the upper and lower bounds (2.4.20) for each node after 5 steps of the algorithm. The graphs on the right-hand side display the analogous bounds obtained from (2.4.21). It is clear that the bounds (2.4.20) are tighter than the bounds obtained from (2.4.21).

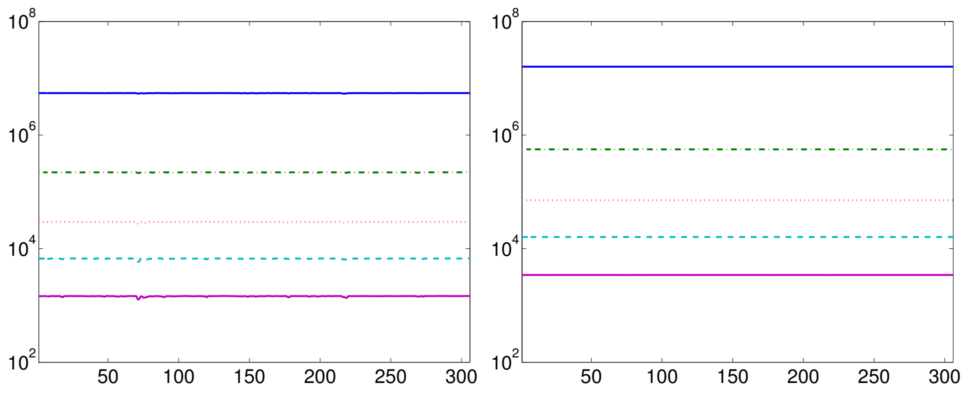


Figure 2.13: Differences between upper and lower bounds during the first 5 steps of Algorithm 3–4. On the left, we report the differences for the bounds (2.4.20), on the right those resulting from (2.4.21).

Table 2.17: Differences between the ranking produced by the hub/authority centrality and by the HITS algorithm. We report the number of nodes, among the first 100, which are placed in a different position by the two methods, and the index of the first different node in each ranking list.

	HITS-hubs		HITS-auth	
matrix	fail	index	fail	index
Celegans	62	12	56	15
PGP	54	35	54	41
Wikipedia	5	34	0	0
Slashdot	6	33	0	0

We conclude this section with a comparison of our approach to the HITS algorithm by Kleinberg [82], which is a popular method for ranking nodes in a directed network. This algorithm gives nodes a large hub score if they point to many important nodes (authorities), and a large authority score if they are pointed to by many important nodes (hubs). It is easy to see that the HITS method is equivalent to only considering the first singular triplet in (2.4.5) and rank the nodes according to the values of the entries of the singular vectors $\mathbf{u}_1$ (hub score) and $\mathbf{v}_1$ (authority score). Therefore, the HITS algorithm may produce rankings that are different from those obtained by evaluating hub and authority centralities. However, Table 2.12 illustrated that only a few singular triplets suffice to identify the most important nodes by using hub and authority centralities. It is therefore interesting to investigate how different the orderings determined by the HITS algorithm and by our approach are. To gain some insight into the orderings produced, we determine the 100 nodes with the largest hub and authority centralities, and compute the 100 most important hubs and authorities with the HITS algorithm for our test networks. We found that the identification and ordering of the 100 most important hubs and authorities obtained by these methods for the networks Airlines, Air500, Twitter, and Wikivote are the same. The orderings differ for the networks of Table 2.17. The second and fourth columns of the table show how many nodes among the 100 nodes in each list differ; the third and fifth columns show the index of the first node that differs in each list. The table illustrates that the HITS algorithm does not always yield the nodes with the largest hub/authority centrality. Moreover, when a few singular triplets of the adjacency matrix are sufficient to determine an accurate ordering of the nodes hub and authority centralities, the HITS algorithm does not have a significant advantage in terms of complexity over our method.

Application of block algorithms This subsection presents computations that illustrate the performance of block Gauss and anti-Gauss quadrature rules associated with the nonsymmetric block Lanczos algorithm. We applied these quadrature rules to eight directed unweighted networks coming from the following real-world applications:

Airlines (235 nodes, 2101 edges) is a representation of air traffic, available at [55]. The nodes are airports and the directed edges are flights between them.

Celegans (306 nodes, 2345 edges) is the metabolic network of *Caenorhabditis elegans* [38], a small nematode (roundworm). The data set is available at [1].

Air500 (500 nodes, 24009 edges) is the network of flight connections for the top 500 airports, based on total passenger volume, worldwide [14]. The existence of flight connections between airports is based on flights within one year from July 1, 2007, to June 30, 2008 [97].

Twitter (3556 nodes, 188712 edges) is part of the Twitter network [55]. The nodes are users and the directed edges are mentions and retweets between them.

T2 (9801 nodes, 87025 edges) describes a mesh for a nonlinear diffusion problem, taken from the University of Florida Sparse Matrix Collection [125].

Wikipedia (49728 nodes, 941425 edges) is the structure of Italian Wikipedia. In this graph the nodes are plain articles and the links represent references to other articles. It can be downloaded from [93].

Poisson (85623 nodes, 2374949 edges) is a sparse matrix describing a problem in computational fluid dynamics from the University of Florida Sparse Matrix Collection [125].

Vfem (93476 nodes, 1434636 edges) is a vector finite element complex matrix from a problem in electromagnetics [125]. When computing with this matrix, transposition is replaced by transposition and complex conjugation. Inner products also require complex conjugation.

Thus, all matrices except for T2, Poisson, and Vfem are adjacency matrices. When applying the nonsymmetric block Lanczos algorithm to an arbitrary dense matrix breakdown rarely occurs. However, when A is a general large and sparse nonsymmetric adjacency matrix and the initial vectors are axis vectors $\boldsymbol{e}_i$, and, therefore, very sparse, chances of breakdown are high. For instance, if the matrix A only has a few nonzero entries in each row and column, even though the matrices S_1 and R_1 in (1.4.12) are likely to be of full rank, they almost surely will satisfy $S_1^T R_1 = O_k$, resulting in serious breakdown at the first step. The reason for this difficulty is that independent high-dimensional vectors with only a few nonzero entries are likely to be orthogonal. We remark that this problem does not occur with the symmetric block Lanczos method, because this method requires only the matrix R_1 to be of full rank. The probability of breakdown during the first steps of the nonsymmetric Lanczos method decreases when introducing an additional dense starting vector. This is shown by Bai, Day and Ye [8]; see below for illustrations.

Suppose that we would like to compute all communicabilities between nodes 1 through $k-1$ and their centralities for a total of $(k-1)^2$ numerical quantities. If we try to approximate $\mathcal{I}f = W^T f(A)V$ with $W = V = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{k-1}]$ using the nonsymmetric block Lanczos method with block-size $k-1$, then breakdown is likely to occur at an early stage of the computations. To reduce the likelihood of breakdown, we append the vector $\boldsymbol{c}$ with all entries one to W and V. Thus, we use $W = V = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{k-1}, \boldsymbol{c}]$. Then the computations also provide bounds for the starting and ending conveniences of the first $k-1$ nodes provided that an expansion of the integrand in terms of biorthogonal polynomials converges sufficiently rapidly. The desired f-communicabilities are the entries of the leading principal $(k-1)\times(k-1)$ submatrix of the $k \times k$ matrix $\mathcal{I}f = W^T f(A)V$. We remark that the block Lanczos method is applied to an orthonormalization of the columns in V and W; see below.

Now assume that we would like to compute certain f-communicabilities with a small absolute error τ. Using the quantities T_N, F_N, and G_N in (2.3.14)–(2.3.15), a natural stopping criterion is provided by $T_N < \tau$. We show the performance of block Gauss and anti-Gauss quadrature rules when determining approximations of the entries of $\mathcal{I}f = W^T \exp(A)V$ with $W = V = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_5, \boldsymbol{c}]$ and $\tau = 10^{-3}$.

The initial blocks have to satisfy $V^T W = I_k$. To accomplish this, in principle, we could orthogonalize the columns of the matrix W defined above by a QR factorization, but this would not necessarily change its sparsity pattern and, therefore, would likely lead to breakdown of the nonsymmetric block Lanczos method. In fact, when A and W are sparse, either one of matrices R_1 and S_1 in (1.4.12), or both, may be singular. For this reason, we redefine the blocks W and V with the aid of the singular value decomposition of the $k \times k$ block $W^T V$. If $W^T V = U\Sigma Z^T$, then the matrices

$$W_1 = WU\Sigma^{-1/2}, \qquad V_1 = VZ\Sigma^{-1/2}$$

satisfy $W_1^T V_1 = I_k$ and have dense columns. The sought quantities can be determined from

$$W^T f(A)V = U\Sigma^{1/2}(W_1^T f(A)V_1)\Sigma^{1/2}Z^T.$$

To begin with, we repeated the first experiment of Subsection 2.3.2. Let the matrix W be defined as described above. We computed approximations of $[W^T f(A)W]_{ij}$ by pairs of N-block Gauss and $(N+1)$-block anti-Gauss rules, which were computed with the nonsymmetric block Lanczos method, and compared the 36 entries of the resulting matrices to the output of `expm`. This test was performed on the first four networks, whose size allows the application of `expm`. The inequalities (2.3.16) held in all but a small number of cases: One of the inequalities was violated for 5 (out of 36) entries for the Celegans network and 14 of the entries for the Twitter network. The required accuracy was attained for all examples, that is, the error G_N (2.3.15) was smaller than the stopping tolerance τ for all networks.

Table 2.18 shows the execution times (in seconds) of the MATLAB function `expm` and the nonsymmetric block Lanczos method. The table also reports the number of matrix-vector product evaluations, the number of block Lanczos steps, and the quantity (2.3.15). Very few block Lanczos steps are needed to determine the desired quantities with required accuracy. For the larger networks, we are unable to evaluate the function `expm` and, therefore, the error G_N.

Table 2.18: Execution times (in seconds) of `expm` and the nonsymmetric block Lanczos method. The table also shows the number of matrix-vector product evaluations, the number of block Lanczos steps, and the quantity (2.3.15).

			`expm`	Block Lanczos method			
Matrix	Nodes	Edges	time	Time	MVP	Steps	G_N
Airlines	235	2101	2.4e-01	5.7e-02	66	6	2.8e-08
Celegans	306	2345	1.6e-01	1.7e-02	66	6	8.5e-05
Air500	500	24009	9.8e-02	3.0e-02	66	6	3.1e-08
Twitter	3656	188712	4.4e+02	7.6e-02	90	8	4.1e-13
T2	9801	87025	–	7.4e-02	42	4	–
Wikipedia	49728	941425	–	7.8e-01	90	8	–
Poisson	85623	2374949	–	6.3e-01	30	3	–
Vfem	93476	1434636	–	6.6e-01	18	2	–

3. Electromagnetic Sounding

Electromagnetic induction (EMI) techniques are often used for non-destructive investigation of soil properties, as they are affected by electromagnetic properties of the subsurface layers, namely the *electrical conductivity* σ and the *magnetic permeability* μ. Knowing such parameters allows one to ascertain the presence of particular substances, with many important applications. A *ground conductivity meter* is the basic instrument for EMI. It contains two coils (a transmitter and a receiver) placed at a fixed distance. An alternating sinusoidal current in the transmitter produces a primary magnetic field H_P, which induces small eddy currents in the subsurface. These currents produce a secondary magnetic field H_S, which is sensed by the receiver. The ratio of the secondary to the primary magnetic fields is then used, along with the instrumental parameters, to estimate electrical properties of the subsurface.

The coils axes can be aligned either vertically or horizontally with respect to the ground surface, producing different measures; see Figure 3.1.

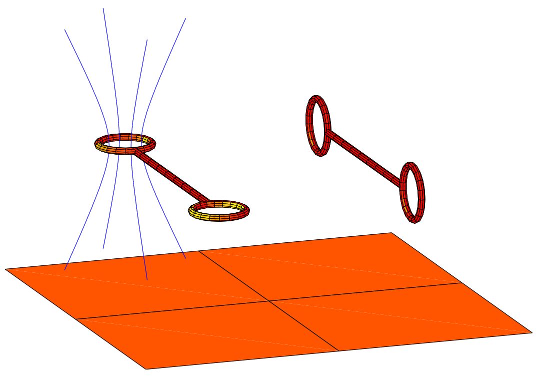


Figure 3.1: Vertical and horizontal alignment of the coils of a GCM

The instruments measures the *apparent conductivity*

$$m = \frac{4\,\mathrm{Im}(H_S/H_P)}{\mu_0 \omega r^2},$$

which coincides with the real conductivity σ under the following assumptions:

- instrument at ground level ($h = 0$), in vertical orientation;
- soil with uniform magnetic permeability $\mu_0 = 4\pi 10^{-7}$ H/m;

- soil with uniform electrical conductivity σ;
- small *induction number*

$$B = \frac{r}{\delta} = r\sqrt{\frac{1}{2}\mu_0\omega\sigma} \ll 1, \tag{3.0.1}$$

where r is the inter-coil distance, δ is the *skin depth* (attenuation of H_P by a factor e^{-1}) and $\omega = 2\pi f$, where f is the operating frequency.

In real applications the assumption of uniform soil conductivity is not realistic. Moreover, geophysicists are particularly interested in non homogeneous soil and, in addition, apparent conductivity gives no information on the depth localization of inhomogeneities.

To face the problem of data inversion multiple measures are needed to recover the distribution of conductivity with respect to depth. Among the parameters which influence the GCM responses, we have been able to generate multiple measures letting the orientation of the dipole change and taking measurements at different height over the ground.

In 1980, McNeill [101] described a **linear model**, based on the *response curves* in the vertical and horizontal positions, which relates the apparent conductivity to the height over the ground

$$m^V(h) = \int_0^\infty \phi^V(h+z)\sigma(z)\,dz$$
$$m^H(h) = \int_0^\infty \phi^H(h+z)\sigma(z)\,dz$$

where $\sigma(z)$ is the real conductivity,

$$\phi^V(z) = \frac{4z}{(4z^2+1)^{3/2}}, \qquad \phi^H(z) = 2 - \frac{4z}{(4z^2+1)^{1/2}},$$

and z is the ratio between the depth and the inter-coil distance r.

Unfortunately, the linear model is only valid for uniform magnetic permeability μ_0, small induction number B and moderate conductivity ($\sigma \lesssim 100\,\mathrm{mS/m}$).

3.1 The nonlinear forward model

The nonlinear model described in [131, 132], and further analyzed and adapted to the case of a GCM in [74], is derived from Maxwell's equations, keeping in mind the cylindrical symmetry of the problem, due to the magnetic field sensed by the receiver coil being independent of the rotation of the instrument around the vertical axis. The input quantities are the distribution of the electrical conductivity and magnetic permeability in the subsurface, the output is the apparent conductivity at height h. In the following, λ is a variable of integration which has no particular physical meaning. It can be interpreted as the ratio between a length and the *skin depth* δ; see (3.0.1).

Following [131, Chapter III], we assume that the soil has a layered structure with n layers, each of thickness d_k, $k = 1, \dots, n$, and consequently that the electromagnetic variables are piecewise constant; see Figure 3.2. The thickness d_n of the bottom layer is assumed to be infinite. Let σ_k and μ_k be the electrical conductivity and the magnetic permeability of the k-th layer, respectively, and let $u_k(\lambda) = \sqrt{\lambda^2 + \mathrm{i}\sigma_k\mu_k\omega}$, where

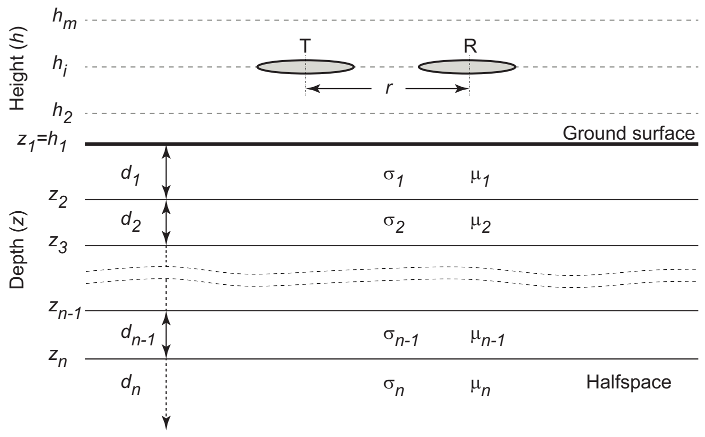


Figure 3.2: Schematic representation of the subsoil and of the discretization used in the following.

$\mathrm{i} = \sqrt{-1}$ is the imaginary unit. Then, the characteristic admittance of the k-th layer is given by

$$N_k(\lambda) = \frac{u_k(\lambda)}{\mathrm{i}\mu_k\omega}, \qquad k = 1, \dots, n. \tag{3.1.1}$$

The surface admittance at the top of the k-th layer is denoted by $Y_k(\lambda)$ and verifies the following recursion

$$Y_k(\lambda) = N_k(\lambda)\frac{Y_{k+1}(\lambda) + N_k(\lambda)\tanh(d_k u_k(\lambda))}{N_k(\lambda) + Y_{k+1}(\lambda)\tanh(d_k u_k(\lambda))}, \quad k = n-1, \dots, 1, \tag{3.1.2}$$

which is initialized by setting $Y_n(\lambda) = N_n(\lambda)$ at the lowest layer. Numerically, this is equivalent to start the recursion at $k = n$ with $Y_{n+1}(\lambda) = 0$.

Now let,

$$R_0(\lambda) = \frac{N_0(\lambda) - Y_1(\lambda)}{N_0(\lambda) + Y_1(\lambda)}, \tag{3.1.3}$$

where $N_0(\lambda) = \lambda/(\mathrm{i}\mu_0\omega)$, and

$$\begin{aligned} T_0(h) &= -\delta^3 \int_0^\infty \lambda^2 e^{-2h\lambda} R_0(\lambda) J_0(r\lambda)\, d\lambda, \\ T_2(h) &= -\delta^2 \int_0^\infty \lambda e^{-2h\lambda} R_0(\lambda) J_1(r\lambda)\, d\lambda, \end{aligned} \tag{3.1.4}$$

where $J_0(\lambda)$ and $J_1(\lambda)$ are Bessel functions of the first kind of order 0 and 1, respectively, and r is the inter-coil distance. We express the integrals (3.1.4) in the variable λ, instead than $g = \delta\lambda$ as in [131]. This has some impact on the numerical computation; see Remark 3.1.1.

The results obtained by Wait in [131, page 113], adapted to the geometry of a GCM, give the components of the magnetic field along the dipole axis

$$\begin{aligned} (H_P)_z &= -\frac{C}{r^3}, \qquad & (H_S)_z &= -\frac{C}{\delta^3} T_0(h), \qquad & &\text{(vertical dipole)}, \\ (H_P)_y &= -\frac{C}{r^3}, \qquad & (H_S)_y &= -\frac{C}{r\delta^2} T_2(h), \qquad & &\text{(horizontal dipole)}, \end{aligned} \tag{3.1.5}$$

where C is a constant.

The apparent conductivity measured by a GCM can be expressed as

$$m = \frac{4}{\mu_0 \omega r^2} \operatorname{Im} \left(\frac{(H_S)_d}{(H_P)_d} \right), \tag{3.1.6}$$

where $(H_P)_d$ and $(H_S)_d$ are the components along the dipole axis of the primary and secondary magnetic field, respectively. Substituting (3.1.5) in (3.1.6), we obtain the predicted values of the apparent conductivity measurement $m^V(h)$ (vertical orientation of coils) and $m^H(h)$ (horizontal orientation of coils) at height h above the ground

$$m^V(h) = \frac{4}{\mu_0 \omega r^2} \operatorname{Im}(B^3 T_0(h)), \qquad m^H(h) = \frac{4}{\mu_0 \omega r^2} \operatorname{Im}(B^2 T_2(h)),$$

where B is the induction number (3.0.1). Simplifying formulae, we find

$$\begin{aligned} m^V(h) &= \frac{4r}{\mu_0 \omega} \mathcal{H}_0 \left[-\lambda e^{-2h\lambda} \operatorname{Im}(R_0(\lambda)) \right](r), \\ m^H(h) &= \frac{4}{\mu_0 \omega} \mathcal{H}_1 \left[-e^{-2h\lambda} \operatorname{Im}(R_0(\lambda)) \right](r). \end{aligned} \tag{3.1.7}$$

Here we denote by

$$\mathcal{H}_\nu[f_h](r) = \int_0^\infty f_h(\lambda) J_\nu(r\lambda) \lambda \, d\lambda \tag{3.1.8}$$

the Hankel transform of order ν of the function $f_h(\lambda)$, where the height h is a fixed parameter. In our numerical experiments we approximate $\mathcal{H}_\nu[f_h](r)$ by the quadrature formula described in [2], using the nodes and weights adopted in [74].

The model depends upon a number of parameters which influence the value of the apparent conductivity. In particular, it is affected by the instrument orientation (horizontal/vertical), its height h above the ground, the inter-coil distance r, and the angular frequency ω. In view of the technical features of the GCM at our disposal, we consider r and ω to be constant. This constraint could be easily removed.

Remark 3.1.1. *The above relations* (3.1.7) *show that the apparent conductivity predicted by the model does not depend explicitly of the* skin depth δ *and the* induction number B. *This has some relevance in numerical computation, as an estimate of the value of* δ *is not required. To our knowledge, this is the first time that this is noted.*

3.2 Solution of the inverse problem

In our analysis, we let the magnetic permeability take the same value μ_0 in the n layers. This assumption is approximately met if the ground does not contain ferromagnetic materials. Then, we can consider the apparent conductivity as a function of the value σ_k of the conductivity in each layer and of the height h, and we write $m^V(\boldsymbol{\sigma}, h)$ and $m^H(\boldsymbol{\sigma}, h)$, where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)^T$, instead than $m^V(h)$ and $m^H(h)$.

The problem of data inversion is very important in Geophysics, when one is interested in depth localization of inhomogeneities of the soil. To this purpose, multiple measurements are needed to recover the distribution of conductivity with respect to depth. In order to obtain such measurements, we use the two admissible loop orientations and assume to record apparent conductivity at height h_i, $i = 1, \ldots, m$, as depicted in Figure 3.2. This generates $2m$ data values. The algorithm could be easily adapted to the case when other parameters of the model are varied.

Now, let b_i^V and b_i^H be the data recorded by the GCM at height h_i in the vertical and horizontal orientation, respectively, and let us denote by $r_i(\boldsymbol{\sigma})$ the error in the model prediction for the ith observation

$$r_i(\boldsymbol{\sigma}) = \begin{cases} b_i^V - m^V(\boldsymbol{\sigma}, h_i), & i = 1, \dots, m, \\ b_{i-m}^H - m^H(\boldsymbol{\sigma}, h_{i-m}), & i = m+1, \dots, 2m. \end{cases} \tag{3.2.1}$$

Setting $\mathbf{b}^V = (b_1^V, \dots, b_m^V)^T$, $\mathbf{m}^V(\boldsymbol{\sigma}) = (m^V(\boldsymbol{\sigma}, h_1), \dots, m^V(\boldsymbol{\sigma}, h_m))^T$, and defining $\mathbf{b}^H$ and $\mathbf{m}^H(\boldsymbol{\sigma})$ similarly, we can write the residual vector, the measured data, and the model predictions as

$$\mathbf{r}(\boldsymbol{\sigma}) = \mathbf{b} - \mathbf{m}(\boldsymbol{\sigma}), \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}^V \\ \mathbf{b}^H \end{bmatrix}, \quad \mathbf{m}(\boldsymbol{\sigma}) = \begin{bmatrix} \mathbf{m}^V(\boldsymbol{\sigma}, \mathbf{h}) \\ \mathbf{m}^H(\boldsymbol{\sigma}, \mathbf{h}) \end{bmatrix}. \tag{3.2.2}$$

The problem of data inversion consists of computing the conductivity σ_k of each layer ($k = 1, \dots, n$) which determines a given data set $\mathbf{b} \in \mathbb{R}^{2m}$. As it is customary, we use a least squares approach by solving the nonlinear problem

$$\min_{\boldsymbol{\sigma} \in \mathbb{R}^n} f(\boldsymbol{\sigma}), \qquad f(\boldsymbol{\sigma}) = \frac{1}{2} \|\mathbf{r}(\boldsymbol{\sigma})\|^2 = \frac{1}{2} \sum_{i=1}^{2m} r_i^2(\boldsymbol{\sigma}), \tag{3.2.3}$$

where $\|\cdot\|$ denotes the Euclidean norm and $r_i(\boldsymbol{\sigma})$ is defined in (3.2.1).

To estimate the computational complexity needed to evaluate $\mathbf{r}(\boldsymbol{\sigma})$ we assume that the complex arithmetic operations are implemented according to the classical definitions, i.e., that 2 floating point operations (*flops*) are required for each complex sum, 6 for each product and 11 for each division. The count of other functions (exponential, square roots, etc.) is given separately. If n is the number of layers, $2m$ the number of data values, and q the nodes in the quadrature formula used to approximate (3.1.8), we obtain a complexity $O((45n + 8m)q)$ *flops* plus $2nq$ evaluations of functions with a complex argument, and mq with a real argument.

3.2.1 Inversion algorithm

The classical approach for solving (3.2.3) is to find a stationary point of the gradient $\mathbf{f}'(\boldsymbol{\sigma})$ of $f(\boldsymbol{\sigma})$ by Newton's method, as described in Subsection 1.6.2. Using the notation of this chapter, the iterative step $\mathbf{s}_k$ is chosen by solving the $n \times n$ linear system

$$\mathbf{f}''(\boldsymbol{\sigma}_k)\mathbf{s}_k = -\mathbf{f}'(\boldsymbol{\sigma}_k),$$

where $\mathbf{f}''(\boldsymbol{\sigma}_k)$ is the Hessian of $f(\boldsymbol{\sigma})$. While $\mathbf{f}'(\boldsymbol{\sigma})$ can be obtained analytically or in an approximated way, as we will see in the next Subsection, the analytical expression of $\mathbf{f}''(\boldsymbol{\sigma})$ is not available; it could be computed by further differentiating the gradient, but this would imply a large computational cost. To overcome this difficulty we resort to the Gauss–Newton method, which minimizes at each step the norm of a linear approximation of the residual $\mathbf{r}(\boldsymbol{\sigma} + \mathbf{s})$; see Subsection 1.6.2.

Let $\mathbf{r}(\boldsymbol{\sigma})$ be Fréchet differentiable and $\boldsymbol{\sigma}_k$ denote the current approximation, then we can write

$$\mathbf{r}(\boldsymbol{\sigma}_{k+1}) \simeq \mathbf{r}(\boldsymbol{\sigma}_k) + J(\boldsymbol{\sigma}_k)\mathbf{s}_k,$$

where $\boldsymbol{\sigma}_{k+1} = \boldsymbol{\sigma}_k + \mathbf{s}_k$ and $J(\boldsymbol{\sigma})$ is the $2m \times n$ Jacobian of $\mathbf{r}(\boldsymbol{\sigma})$, defined by (1.6.6).

As we saw in the first chapter, at each step k, $\mathbf{s}_k$ is the solution of the linear least squares problem

$$\min_{\mathbf{s}\in\mathbb{R}^n} \|\mathbf{r}(\boldsymbol{\sigma}_k) + J_k\mathbf{s}\|, \tag{3.2.4}$$

and from it we obtain the iterative method

$$\boldsymbol{\sigma}_{k+1} = \boldsymbol{\sigma}_k + \mathbf{s}_k = \boldsymbol{\sigma}_k - J_k^\dagger\,\mathbf{r}(\boldsymbol{\sigma}_k), \tag{3.2.5}$$

where $J_k^\dagger$ is the Moore–Penrose pseudoinverse of J_k.

When the residuals $r_i(\boldsymbol{\sigma}_k)$ are small or mildly nonlinear at $\boldsymbol{\sigma}_k$, the Gauss–Newton method is expected to behave similarly to Newton's method [16, Chapter 9.2.2]. We remark that, while the physical problem is obviously consistent, this is not necessarily true in our case, where the conductivity $\sigma(z)$ is approximated by a piecewise constant function. Furthermore, in the presence of noise in the data the problem will certainly be inconsistent. At the same time, since we are focused on the nonlinear case, connected to the presence of strong conductors in the subsoil, we do not take into account the second possibility. We remark that in the case of a mildly nonlinear problem, a linear model is available [20, 101]. If the above conditions are not satisfied, the Gauss–Newton method may not converge.

For this reason, we replaced the approximation (3.2.5) by

$$\boldsymbol{\sigma}_{k+1} = \boldsymbol{\sigma}_k + \alpha_k\mathbf{s}_k, \tag{3.2.6}$$

that is, using the damped Gauss–Newton method that we described in Subsection 1.6.2. This choice of α_k ensures convergence of the method, provided that $\boldsymbol{\sigma}_k$ is not a critical point [16, Chapter 9.2.1].

The damped method allows us to include an important physical constraint in the inversion algorithm, i.e., the positivity of the solution. In our implementation α_k is the largest step size which both satisfies the Armijo–Goldstein principle and ensures that all the solution components are positive.

3.2.2 Computation of the Jacobian

As we saw in the previous Subsection, being able to compute or to approximate the Jacobian matrix $J(\boldsymbol{\sigma})$ of the vector function (3.2.2) is crucial for the implementation of an effective inversion algorithm and to have information about its speed of convergence and conditioning.

The classical approach is to resort to a finite difference approximation

$$\frac{\partial r_i(\boldsymbol{\sigma})}{\partial \sigma_j} \simeq \frac{r_i(\boldsymbol{\sigma}+\boldsymbol{\delta}_j) - r_i(\boldsymbol{\sigma})}{\delta}, \quad i = 1,\dots,2m,\ j = 1,\dots,n, \tag{3.2.7}$$

where $\boldsymbol{\delta}_j = \delta\,\mathbf{e}_j = (0,\dots,0,\delta,0,\dots,0)^T$ and δ is a fixed constant; see [74].

In this section we give the explicit expression of the Jacobian matrix. We will show that the complexity of this computation is smaller than required by the finite difference approximation (3.2.7). In the following lemma we omit, for clarity, the variable λ.

Lemma 3.2.1. *The derivatives* $Y'_{kj} = \frac{\partial Y_k}{\partial \sigma_j}$, $k, j = 1,\dots,n$, *of the surface admittances* (3.1.2) *can be obtained starting from*

$$Y'_{nn} = \frac{1}{2u_n}, \qquad Y'_{nj} = 0, \qquad j = 1,\dots,n-1, \tag{3.2.8}$$

and proceeding recursively for $k = n-1, n-2, \dots, 1$ by

$$\begin{aligned} Y'_{kj} &= N_k^2 b_k Y'_{k+1,j}, \qquad j = n, n-1, \dots, k+1, \\ Y'_{kk} &= \frac{a_k}{2u_k} + \frac{b_k}{2}\left[N_k^2 d_k - Y_{k+1}\left(d_k Y_{k+1} + \frac{1}{\mathrm{i}\mu_k\omega}\right)\right], \\ Y'_{kj} &= 0, \qquad j = k-1, k-2, \dots, 1, \end{aligned} \tag{3.2.9}$$

where

$$a_k = \frac{Y_{k+1} + N_k \tanh(d_k u_k)}{N_k + Y_{k+1}\tanh(d_k u_k)}, \qquad b_k = \frac{1}{[N_k + Y_{k+1}\tanh(d_k u_k)]^2 \cosh^2(d_k u_k)}. \tag{3.2.10}$$

Proof. From (3.1.1) we obtain

$$\frac{\partial u_k}{\partial \sigma_j} = \frac{\partial}{\partial \sigma_j}\sqrt{\lambda^2 + \mathrm{i}\sigma_k\mu_k\omega} = \frac{1}{2N_k}\delta_{kj}, \qquad \frac{\partial N_k}{\partial \sigma_j} = \frac{\partial}{\partial \sigma_j}\frac{u_k}{\mathrm{i}\mu_k\omega} = \frac{1}{2u_k}\delta_{kj}, \tag{3.2.11}$$

where δ_{kj} is the Kronecker delta, that is, 1 if $k = j$ and 0 otherwise. The recursion initialization (3.2.8) follows from $Y_n = N_n$; see Section 3.1. We have

$$\begin{aligned} Y'_{kj} = \frac{\partial N_k}{\partial \sigma_j} a_k + N_k \cdot \frac{\frac{\partial Y_{k+1}}{\partial \sigma_j} + \frac{\partial N_k}{\partial \sigma_j}\tanh(d_k u_k) + N_k \frac{\partial \tanh(d_k u_k)}{\partial \sigma_j}}{N_k + Y_{k+1}\tanh(d_k u_k)} \\ - N_k a_k \cdot \frac{\frac{\partial N_k}{\partial \sigma_j} + \frac{\partial Y_{k+1}}{\partial \sigma_j}\tanh(d_k u_k) + Y_{k+1}\frac{\partial \tanh(d_k u_k)}{\partial \sigma_j}}{N_k + Y_{k+1}\tanh(d_k u_k)}, \end{aligned}$$

with a_k defined as in (3.2.10). If $j \neq k$, then $\frac{\partial N_k}{\partial \sigma_j} = \frac{\partial u_k}{\partial \sigma_j} = 0$ and we obtain

$$Y'_{kj} = N_k^2 \frac{\frac{\partial Y_{k+1}}{\partial \sigma_j}\left(1 - \tanh^2(d_k u_k)\right)}{[N_k + Y_{k+1}\tanh(d_k u_k)]^2} = N_k^2 b_k Y'_{k+1,j}.$$

The last formula, with b_k given by (3.2.10), avoids the cancellation in $1 - \tanh^2(d_k u_k)$.

If $j = k$, after some straightforward simplifications, we get

$$\begin{aligned} Y'_{kk} = \frac{\partial N_k}{\partial \sigma_k} a_k + \frac{N_k}{N_k + Y_{k+1}\tanh(d_k u_k)}\Bigg[Y'_{k+1,k}(1 - a_k \tanh(d_k u_k)) \\ + \frac{\partial N_k}{\partial \sigma_k}(\tanh(d_k u_k) - a_k) + \frac{d_k}{2}\left(1 - a_k \frac{Y_{k+1}}{N_k}\right)(1 - \tanh^2(d_k u_k))\Bigg]. \end{aligned}$$

This formula, using (3.2.10) and (3.2.11), leads to

$$Y'_{kk} = \frac{a_k}{2u_k} + N_k b_k\left[N_k\left(Y'_{k+1,k} + \frac{d_k}{2}\right) - \frac{1}{2}Y_{k+1}\left(\frac{d_k}{N_k}Y_{k+1} + \frac{1}{u_k}\right)\right].$$

The initialization (3.2.8) implies that $Y'_{kj} = 0$ for any $j < k$. In particular $Y'_{k+1,k} = 0$, and since N_k/u_k is constant one obtains the expression of Y'_{kk} given in (3.2.9). This completes the proof. □

Remark 3.2.1. *The quantity a_k in (3.2.10) appears in the right hand side of (3.1.2), and its denominator is present also in the expression of b_k. It is therefore possible to implement jointly the recursions (3.1.2) and (3.2.9) in order to reduce the number of floating point operations required by the computation of the Jacobian. We also note that, since in the following Theorem 3.2.1 we only need the partial derivatives of Y_1, we can overwrite the values of $Y'_{k+1,j}$ with Y'_{kj} at each recursion step, so that only n storage locations are needed for each λ value, instead of n^2.*

Theorem 3.2.1. *The partial derivatives of the residual function* (3.2.2) *are given by*

$$\frac{\partial r_i(\boldsymbol{\sigma})}{\partial \sigma_j} = \begin{cases} \frac{4r}{\mu_0\omega}\mathcal{H}_0\left[\lambda e^{-2h_i\lambda}\,\mathrm{Im}\left(\frac{\partial R_0(\lambda)}{\partial \sigma_j}\right)\right](r), & i = 1,\ldots,m, \\ \\ \frac{4}{\mu_0\omega}\mathcal{H}_1\left[e^{-2h_{i-m}\lambda}\,\mathrm{Im}\left(\frac{\partial R_0(\lambda)}{\partial \sigma_j}\right)\right](r), & i = m+1,\ldots,2m, \end{cases}$$

for $j = 1,\ldots,n$. *Here* $\mathcal{H}_\nu$ $(\nu = 0,1)$ *denotes the Hankel transform* (3.1.8), r *is the inter-coil distance,* $\frac{\partial R_0(\lambda)}{\partial \sigma_j}$ *is the* j*th component of the gradient of the function* (3.1.3)

$$\frac{\partial R_0(\lambda)}{\partial \sigma_j} = \frac{-2\mathrm{i}\mu_0\omega\lambda}{(\lambda + \mathrm{i}\mu_0\omega Y_1(\lambda))^2}\cdot\frac{\partial Y_1}{\partial \sigma_j},$$

and the partial derivatives $\frac{\partial Y_1}{\partial \sigma_j}$ *are given by Lemma 3.2.1.*

Proof. The proof follows easily from Lemma 3.2.1 and from equations (3.1.3), (3.1.7), and (3.2.1). □

Remark 3.2.2. *The numerical implementation of the above formulae needs care. It has already been noted in the proof of Lemma 3.2.1 that equations* (3.2.9)–(3.2.10) *are written in order to avoid cancellations that may introduce huge errors in the computation. Moreover, to prevent overflow in the evaluation of the term*

$$\cosh^2(d_k u_k(\lambda)) = \cosh^2(d_k\sqrt{\lambda^2 + \mathrm{i}\sigma_k\mu_k\omega})$$

in the denominator of b_k*, we fix a value* λ_{max} *and for* $\mathrm{Re}(d_k u_k(\lambda)) > \lambda_{max}$ *we let* $b_k = b_k(\lambda) = 0$. *In our numerical experiments we adopt the value* $\lambda_{max} = 300$.

The complexity of the joint computation of $\mathbf{r}(\boldsymbol{\sigma})$ and its Jacobian, given in Theorem 3.2.1, amounts to $O((3n^2+8mn)q)$ *flops*, $3nq$ complex functions, and mnq real functions. To approximate the Jacobian by finite differences one has to evaluate $n+1$ times $\mathbf{r}(\boldsymbol{\sigma})$, corresponding to $O((45n^2+8mn)q)$ *flops*, $2n^2q$ complex functions, and mnq real functions.

If the Jacobian is a square matrix, i.e., $n = 2m$, its computation is 7 times faster than approximating it by finite differences. The situation improves when the dimension of the data set is smaller than the number of layers (this is the ideal situation, as it will be shown in Sections 3.3.1 and 3.3.2), e.g., the speedup factor is 9 for $n = 4m$. The complexity issue is of concern to end users, because it is often desirable to process the field data in real time, during the measurement campaign, using a notebook computer.

In order to further reduce the computational cost, it is possible to resort to the *Broyden update* of the Jacobian [22], which can be interpreted as a generalization of the secant method. The procedure consists of updating an initial approximation of the Jacobian $J_0 = J(\boldsymbol{\sigma}_0)$ computed in the initial point $\boldsymbol{\sigma}_0$. This is realized by the following rank-1 update

$$J_k = J_{k-1} + \frac{(\mathbf{y}_k - J_{k-1}\mathbf{s}_k)\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{s}_k}, \qquad k = 1,2,\ldots, \tag{3.2.12}$$

where $\mathbf{s}_k = \boldsymbol{\sigma}_k - \boldsymbol{\sigma}_{k-1}$ and $\mathbf{y}_k = r(\boldsymbol{\sigma}_k) - r(\boldsymbol{\sigma}_{k-1})$. This formula makes the linearization $r(\boldsymbol{\sigma}_k) + J_k(\boldsymbol{\sigma} - \boldsymbol{\sigma}_k)$ exact in $\boldsymbol{\sigma}_{k-1}$ and guarantees the least change in the Frobenius norm $\|J_k - J_{k-1}\|_F$. As this method works well locally [37, Chapter 8], in the sense that the accuracy of the approximation degrades as the iteration index grows, we apply recursion (3.2.12) for $k = 1, \dots, k_B - 1$, and we reinitialize the method with the exact Jacobian after k_B iterations. A single application of (3.2.12) takes $10mn + 2(m+n)$ *flops*, to be added to the cost of the evaluation of $\mathbf{r}(\boldsymbol{\sigma})$. We will investigate the performance of this method in Section 3.3.1.

3.2.3 Low-rank approximation as regularization method

With the aim of investigating the conditioning of problem (3.2.3), we examined the numerical behavior of the singular values of the Jacobian matrix $J = J(\boldsymbol{\sigma})$ of the vector function $\mathbf{r}(\boldsymbol{\sigma})$. Let $J = U\Gamma V^T$ be the singular value decomposition (SVD) [16] of the Jacobian, where U and V are orthogonal matrices of size $2m$ and n, respectively, $\Gamma = \text{diag}(\gamma_1, \dots, \gamma_p, 0, \dots, 0)$ is the diagonal matrix of the singular values, and p is the rank of J. We recall that the condition number of J is given by γ_1/γ_p.

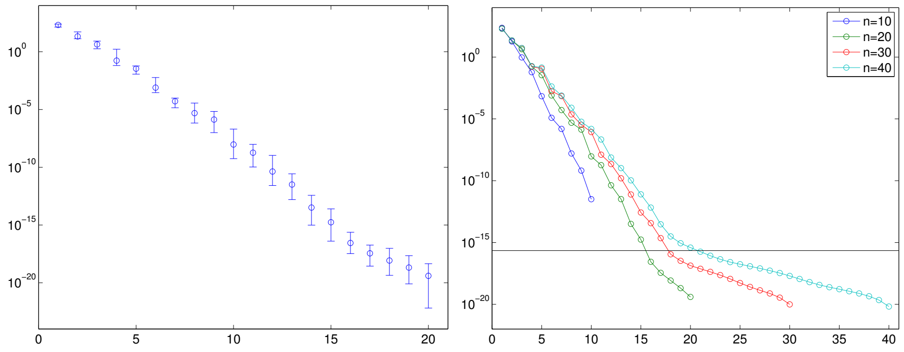


Figure 3.3: SVD of the Jacobian matrix: left, average singular values and errors ($n = 20$); right, average singular values for $n = 10, 20, 30, 40$.

Fixed $n = 2m = 20$, we generate randomly 1000 vectors $\boldsymbol{\sigma} \in \mathbb{R}^{20}$, having components in $[0, 100]$. For each of them we evaluate the corresponding Jacobian $J(\boldsymbol{\sigma})$ by the formulae given in Theorem 3.2.1 and compute its SVD. The left graph in Figure 3.3 shows the average of the singular values obtained by the above procedure and, for each of them, its minimum and maximum value. It is clear that the deviation from the average is small, so that the condition number of the Jacobian matrix has the same order of magnitude in all tests. Consequently, the linearized problem is severely ill-conditioned independently of the value of $\boldsymbol{\sigma}$, and we do not expect its condition number to change much during iteration.

The graph on the right in Figure 3.3 reports the average singular values when $n = 2m = 10, 20, 30, 40$. The figure shows that the condition number is about 10^{14} when $n = 10$ and increases with the dimension. The singular values appear to be exponentially decaying, but zero is not a singular value, that is, the problem is not strictly rank-deficient. The decay rate of the computed singular values changes below machine precision $(2.2 \cdot 10^{16})$, which is represented in the graph by a horizontal line. The exact singular vales are likely to decay with a stronger rate, while the computed ones are probably significantly perturbated by error propagation. A problem of this kind is generally referred to as a *discrete ill-posed problem* [69].

A typical approach for the solution of ill-posed problems is Tikhonov regularization. It has been applied by various author to the inversion of geophysical data; see, e.g., [20, 35, 74]. To apply Tikhonov's method to the nonlinear problem (3.2.3), one has to solve the minimization problem

$$\min_{\boldsymbol{\sigma}\in\mathbb{R}^{\mathbf{n}}} \{\|\mathbf{r}(\boldsymbol{\sigma})\|^2 + \mu^2\|M\boldsymbol{\sigma}\|^2\} \tag{3.2.13}$$

for a fixed value of the parameter μ, where M is a regularization matrix which is often chosen as the identity matrix, or a discrete approximation of the first or second derivatives, as explained in Subsection 1.6.3. In general, choosing the regularization parameter requires the computation of the solution $\boldsymbol{\sigma}_\mu$ of (3.2.13) for many values of μ. This can be done, for example, by the Gauss–Newton method, leading to a large computational effort.

To reduce the complexity we consider an alternative regularization technique based on a low-rank approximation of the Jacobian matrix, i.e., the TSVD approach described in Subsection 1.6.3. If A_ℓ is the best rank ℓ approximation to the Jacobian, then the corresponding solution to (3.2.4) can be expressed as

$$\mathbf{s}^{(\ell)} = -A_\ell^\dagger\mathbf{r} = -\sum_{i=1}^{\ell} \frac{\mathbf{u}_i^T\mathbf{r}}{\gamma_i}\mathbf{v}_i, \tag{3.2.14}$$

where $\ell = 1, \dots, p$ is the regularization parameter, γ_i are the singular values, the singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are the orthogonal columns of U and V, respectively, and $\mathbf{r} = \mathbf{r}(\boldsymbol{\sigma}_k)$.

Introducing a regularization matrix $M \in \mathbb{R}^{t\times n}$ $(t \leq n)$, problem (3.2.4) is usually replaced by

$$\min_{\mathbf{s}\in\mathcal{S}} \|M\mathbf{s}\|, \qquad \mathcal{S} = \{\mathbf{s}\in\mathbb{R}^n \ : \ J^TJ\mathbf{s} = -J^T\mathbf{r}\}, \tag{3.2.15}$$

under the assumption $\mathcal{N}(J)\cap\mathcal{N}(M) = \{0\}$ and $t > \max(0, n-2m)$.

As we saw, the truncated GSVD (TGSVD) solution $\mathbf{s}_\ell$ to (3.2.15) is then defined as

$$\mathbf{s}^{(\ell)} = -\sum_{i=\overline{p}-\ell+1}^{\overline{p}} \frac{\mathbf{u}_{2m-p+i}^T\mathbf{r}}{c_i}\mathbf{z}_{n-p+i} - \sum_{i=\overline{p}+1}^{p} (\mathbf{u}_{2m-p+i}^T\mathbf{r})\,\mathbf{z}_{n-p+i}, \tag{3.2.16}$$

where $\ell = 0, 1, \dots, \overline{p}$ is the regularization parameter, $\overline{p} = t$ if $2m \geq n$, and $\overline{p} = 2m-n+t$ if $2m < n$. We recall that c_i $(i = 1, \dots, \overline{p})$ are the elements of Σ_J different from 0 and 1.

Our approach for constructing a smooth solution to (3.2.3) consists of regularizing each step of the damped Gauss–Newton method (3.2.6) by either TSVD or TGSVD, depending on the choice of M. For a fixed value of the regularization parameter ℓ, we substitute $\mathbf{s}$ in (3.2.6) by $\mathbf{s}^{(\ell)}$ expressed by either (3.2.14) or (3.2.16). We let the resulting method

$$\boldsymbol{\sigma}_{k+1}^{(\ell)} = \boldsymbol{\sigma}_k^{(\ell)} + \alpha_k\mathbf{s}_k^{(\ell)} \tag{3.2.17}$$

iterate until

$$\|\boldsymbol{\sigma}_k^{(\ell)} - \boldsymbol{\sigma}_{k-1}^{(\ell)}\| < \tau\|\boldsymbol{\sigma}_k^{(\ell)}\| \quad \text{or} \quad k > 100 \quad \text{or} \quad \alpha_k < 10^{-5},$$

for a given tolerance τ. The constraint on α_k is a failure condition which indicates that the method does not converge to a positive solution. This typically happens when the solution blows up because of ill-conditioning. We denote the solution at convergence by $\boldsymbol{\sigma}^{(\ell)}$. We will discuss the choice of ℓ in the next subsection.

3.2.4 Choice of the regularization parameter

In the previous section we saw how to regularize the ill-conditioned problem (3.2.3) with the aid of T(G)SVD. The choice of the regularization parameter is crucial in order to obtain a good approximation $\boldsymbol{\sigma}^{(\ell)}$ of $\boldsymbol{\sigma}$.

In real-world applications experimental data are always affected by noise. To model this situation, we assume that the data vector in the residual function (3.2.2), whose norm is minimized in problem (3.2.3), can be expressed as $\mathbf{b} = \widehat{\mathbf{b}} + \mathbf{e}$, where $\widehat{\mathbf{b}}$ contains the exact data and $\mathbf{e}$ is the noise vector. This vector is generally assumed to have normally distributed entries with mean zero and common variance. In real data sets the last condition is not necessarily met.

If an accurate estimate of the norm of the error $\mathbf{e}$ is known, the value of ℓ can be determined with the aid of the discrepancy principle [41, Section 4.3]. It consists of determining the regularization parameter ℓ as the smallest index $\ell = \ell_{\text{discr}}$ such that

$$\|\mathbf{b} - \mathbf{m}(\boldsymbol{\sigma}_{\ell_{\text{discr}}})\| \leq \kappa \|\mathbf{e}\|. \tag{3.2.18}$$

Here $\kappa > 1$ is a user-supplied constant independent of $\|\mathbf{e}\|$. In our experiments we set $\kappa = 1.5$, since it produced the best numerical results.

We are also interested in the situation when an accurate bound for $\|\mathbf{e}\|$ is not available and, therefore, the discrepancy principle cannot be applied.

We can then resort to the heuristic methods described in Subsection 1.6.3. In general, it is not possible to apply all the methods developed for the linear case to a nonlinear problem. The L-curve criterion [73], can be extended quite naturally to the nonlinear case. The L-curve is obtained by joining the points

$$\left\{ \log \|\mathbf{r}(\boldsymbol{\sigma}^{(\ell)})\|, \log \|M\boldsymbol{\sigma}^{(\ell)}\| \right\}, \quad \ell = 1, \ldots, \overline{p}, \tag{3.2.19}$$

where $\mathbf{r}(\boldsymbol{\sigma}^{(\ell)}) = \mathbf{b} - \mathbf{m}(\boldsymbol{\sigma}^{(\ell)})$ is the residual error associated to the approximate solution $\boldsymbol{\sigma}^{(\ell)}$ computed by the iterative method (3.2.17), using (3.2.16) as a regularization method. If (3.2.14) is used instead, it is sufficient to let $M = I$ and replace $\overline{p}$ by p.

In the following Section we will compare different ways of determining the corner of the L-curve.

3.3 Numerical experiments

3.3.1 Synthetic data

To illustrate the performance of the inversion method described in the previous sections, we present the results of a set of numerical experiments. Initially, we apply our method to synthetic data sets, generated starting from a chosen conductivity distribution and adding random noise to data. In the next section we will analyze a real data set.

Figure 3.4 displays the three functions $f_r(z)$, $r = 1, 2, 3$, used in our experiments to model the distribution of conductivity, expressed in Siemens/meter, with respect to the depth z, measured in meters. The first one is differentiable ($f_1(z) = \mathrm{e}^{-(z-1.2)^2}$), the second is piecewise linear, the third is a step function. All model functions imply the presence of a strongly conductive material at a given depth. We assume that the measurements are taken with the GCM in both vertical and horizontal orientation, placed at height $h_i = (i-1)\bar{h}$ above the ground, $i = 1, \ldots, m$, for a chosen height step $\bar{h}$; see (3.2.1). In our experiments $\bar{h} \geq 0.1$ meters.

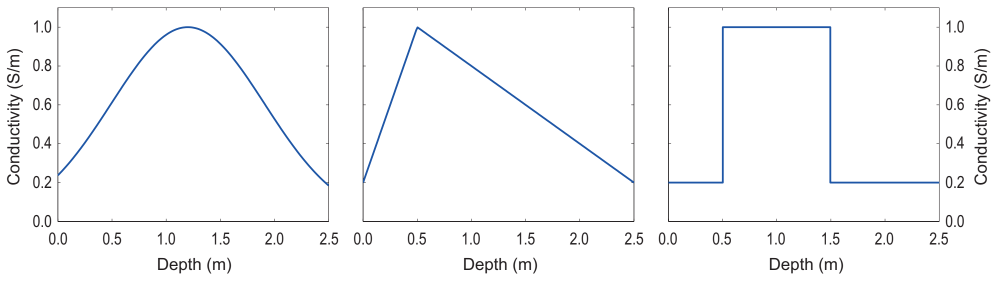


Figure 3.4: Graphs of the conductivity distribution models f_1, f_2, and f_3. The horizontal axis reports the depth in meters, the vertical axis the electrical conductivity in Siemens/meter.

In this section we simulate the use of a Geonics EM38, operating at frequency 14.6 kHz, with 1 m coil separation. For a chosen model function f_r and a fixed number of layers n, we let the layers thickness assume the constant value $d_k = \bar{d} = 2.5/(n-1)$, $k = 1, \dots, n-1$, so that $z_j = (j-1)\bar{d}$, $j = 1, \dots, n$; see Figure 3.2. The choice of $\bar{d}$ is motivated by the common assumption that this kind of GCM can give useful information about the conductivity of the ground up to a depth of 2–3 meters.

We assign to each layer the conductivity $\sigma_k = f_r(z_k)$. Then, we apply the nonlinear model defined in (3.2.2) to compute the exact data vector $\widehat{\mathbf{b}} = \mathbf{m}(\boldsymbol{\sigma})$.

To simulate experimental errors, we determine the perturbed data vector $\mathbf{b}$ by adding a noise vector to $\widehat{\mathbf{b}}$. Specifically, we let the vector $\mathbf{w}$ have normally distributed entries with zero mean and unitary variance, and compute

$$\mathbf{b} = \widehat{\mathbf{b}} + \frac{\tau \|\widehat{\mathbf{b}}\|}{\sqrt{2m}} \mathbf{w}. \tag{3.3.1}$$

This implies that $\|\mathbf{b} - \widehat{\mathbf{b}}\| \approx \tau \|\widehat{\mathbf{b}}\|$. In the computed examples we use the noise levels $\tau = 10^{-3}, 10^{-2}$. Based on our experience, the noise on experimental data is larger than 10^{-1}, but it can be substantially reduced, e.g., by averaging a small number of repeated measurements.

For each data set, we solve the least squares problem (3.2.3) by the damped Gauss–Newton method (3.2.6). The damping parameter is determined by the Armijo–Goldstein principle, modified in order to ensure the positivity of the solution. Each step of the iterative method is regularized by either the TSVD approach (3.2.14), or by TGSVD (3.2.16), for a given regularization matrix M. In our experiments we use both $M = D_1$ and $M = D_2$, the discrete approximations of the first and second derivatives. These two choices for M pose a constraint on the magnitude of the slope and the curvature of the solution, respectively. To assess the accuracy of the computation we use the relative error

$$e_\ell = \frac{\|\boldsymbol{\sigma} - \boldsymbol{\sigma}^{(\ell)}\|}{\|\boldsymbol{\sigma}\|},$$

where $\boldsymbol{\sigma}$ denotes the exact solution of the problem and $\boldsymbol{\sigma}^{(\ell)}$ its regularized solution with parameter ℓ, obtained by (3.2.17). The experiments were performed using Matlab 8.1 (R2013a) on an Intel Core i7/860 computer with 8Gb RAM, running Linux.

Our first experiment tries to determine the best experimental setting, that is, the optimal number of measurements and of underground layers to be considered. At the same time, we investigate the difference between the TSVD (3.2.14) and the TGSVD

Table 3.1: Optimal error e_{opt} for $m = 5, 10, 20$ and $n = 20, 40$, for the TSVD solution ($M = I$) and the TGSVD solution with $M = D_1$ and $M = D_2$. The Jacobian is computed as in Section 3.2.2.

		$M = I$		$M = D_1$		$M = D_2$	
example	m	$n = 20$	$n = 40$	$n = 20$	$n = 40$	$n = 20$	$n = 40$
	5	4.1e-01	3.8e-01	1.8e-01	1.7e-01	2.3e-01	2.9e-01
f_1	10	3.6e-01	3.7e-01	1.4e-01	1.3e-01	1.8e-01	1.6e-01
	20	3.5e-01	3.5e-01	1.5e-01	1.4e-01	1.2e-01	1.3e-01
	5	4.8e-01	4.6e-01	1.3e-01	1.4e-01	2.2e-01	2.6e-01
f_2	10	4.3e-01	4.0e-01	1.2e-01	9.5e-02	1.3e-01	1.9e-01
	20	3.9e-01	3.7e-01	1.1e-01	9.1e-02	1.5e-01	1.4e-01
	5	5.7e-01	5.6e-01	3.9e-01	3.9e-01	4.2e-01	4.1e-01
f_3	10	5.5e-01	5.4e-01	3.6e-01	3.4e-01	3.4e-01	3.2e-01
	20	5.6e-01	5.6e-01	3.5e-01	3.4e-01	2.9e-01	3.3e-01

(3.2.16) approaches, and the effect on the solution of the regularization matrix M. For each of the three test conductivity models, we discretize the soil by 20 or 40 layers, up to the depth of 2.5 meters. We solve the problem after generating synthetic measurements at 5, 10, and 20 equispaced heights up to 1.9 meters. This process is repeated for each regularization matrix. The (exact) Jacobian is computed as described in Section 3.2.2. Table 3.1 reports the values of the relative error $e_{\text{opt}} = \min_\ell e_\ell$, representing the best possible performance of the method. This value is the average over 20 realizations of the noise, with noise level $\tau = 10^{-3}, 10^{-2}$.

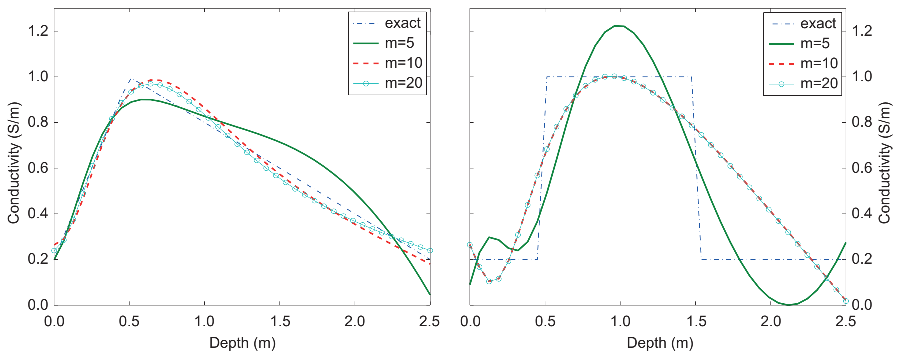


Figure 3.5: Optimal reconstruction for the model functions f_2 and f_3. The number of underground layers is $n = 40$, the noise level is $\tau = 10^{-3}$. The solid line is the solution obtained by taking as input 5 measurements for every loop orientation (that is, $m = 5$), the dashed line corresponds to $m = 10$, the line with bullets to $m = 20$. The exact solution is represented by a dash-dotted line.

It is clear that the TSVD approach (see the column labelled as $M = I$ in the table) is the least accurate. The TGSVD, with either $M = D_1$ or $M = D_2$, gives the best results for the three test functions. The results in Table 3.1 state that they are essentially equivalent, and do not clearly indicate which is the best. We will show in Section 3.3.2 that the regularization matrix $M = D_2$ appears to produce more accurate reconstructions starting from experimental data.

Regarding the size of the soil discretization, it seems convenient to use a large number of layers, that is, $n = 40$. This choice does not increase significantly the computation time. It is obviously desirable to have at disposal a large number of measurements, however the results obtained with $m = 5$ and $m = 10$ are not much worse than those computed with $m = 20$; 5 measurement heights are often sufficient to give a rough approximation of the depth localization of a conductive layer. This is an important remark, as it reduces the time needed for data acquisition.

Figure 3.5 gives an idea of the quality of the computed reconstructions for the model functions f_2 and f_3, with $n = 40$ and noise level $\tau = 10^{-3}$. The exact solution is compared to the approximations corresponding to $m = 5, 10, 20$. The above comments about the influence of the number of measurements m are confirmed. It is also remarkable that the position of the maximum is very well localized.

Table 3.2: Optimal error e_{opt} for $m = 5, 10, 20$ and $n = 20, 40$, for f_1 ($M = D_2$), f_2 ($M = D_1$), and f_3 ($M = D_2$). The results obtained from measurements collected with the instrument in both vertical and horizontal orientation are compared to those obtained with a single orientation.

		f_1, $M = D_2$		f_2, $M = D_1$		f_3, $M = D_2$	
orientation	m	$n = 20$	$n = 40$	$n = 20$	$n = 40$	$n = 20$	$n = 40$
both	5	2.3e-01	2.9e-01	1.3e-01	1.4e-01	4.2e-01	4.1e-01
	10	1.8e-01	1.6e-01	1.2e-01	9.5e-02	3.4e-01	3.2e-01
	20	1.2e-01	1.3e-01	1.1e-01	9.1e-02	2.9e-01	3.3e-01
vertical	5	3.3e-01	2.9e-01	3.5e-01	3.1e-01	6.2e-01	6.6e-01
	10	2.4e-01	1.7e-01	2.9e-01	2.6e-01	5.3e-01	5.0e-01
	20	1.3e-01	2.2e-01	2.4e-01	1.7e-01	4.0e-01	4.3e-01
horizontal	5	2.9e-01	2.7e-01	3.6e-01	3.5e-01	6.6e-01	8.5e-01
	10	2.4e-01	2.6e-01	1.9e-01	1.6e-01	6.3e-01	6.0e-01
	20	2.0e-01	2.1e-01	1.7e-01	1.8e-01	4.4e-01	4.7e-01

In the previous experiments we assumed that all the $2m$ entries of vector $\mathbf{b}$ in (3.2.2) were available. In Table 3.2 we compare these results with those obtained by using only half of them, i.e., those corresponding to either the vertical or horizontal orientation of the instrument. The rows labelled as "both" are extracted from Table 3.1. The results are slightly worse when the number of data is halved, especially for the less regular model functions, while they are almost equivalent for the smooth function f_1. This observation contributes, like the previous one, to simplify and speed up field measurements.

In Section 3.2.2 we described the computation of the Jacobian matrix of (3.2.2), and compared it to the slower finite difference approximation (3.2.7) and to the Broyden update of the Jacobian (3.2.12). To investigate the execution time corresponding to each method, we let the method (3.2.17) perform 100 iterations, with $M = D_2$, for a fixed regularization parameter ($\ell = 4$). When the Jacobian is exactly computed, the execution time is 7.18 s, while the finite difference approximation requires 18.96 s. The speedup factor is 2.6, which is far less than the one theoretically expected. This is probably due to the implementation details and to the features of Matlab programming language. We performed the same experiment by applying the Broyden update (3.2.12) and recomputing the Jacobian every k_B iterations. For $k_B = 5$ the execution time is 2.00 s, while for $k_B = 10$ is 1.32 s. Despite this strong speedup (a factor 14 with respect to finite difference approximation), the accuracy is not substantially affected by this approach. Table 3.3 reports the relative error e_{opt} obtained by repeating the experiment of Table 3.1 using the Broyden method with $k_B = 10$. We only report the values of e_{opt} for some of the examples. The loss of accuracy, if any, is minimal.

Table 3.3: Optimal error e_{opt} for $m = 5, 10, 20$ and $n = 20, 40$, for f_1 ($M = D_2$), f_2 ($M = D_1$), and f_3 ($M = D_2$). The Jacobian is computed every 10 iterations and then updated by the Broyden method.

	f_1, $M = D_2$		f_2, $M = D_1$		f_3, $M = D_2$	
m	$n = 20$	$n = 40$	$n = 20$	$n = 40$	$n = 20$	$n = 40$
5	1.8e-01	2.3e-01	1.3e-01	1.2e-01	4.6e-01	5.0e-01
10	1.7e-01	1.5e-01	1.1e-01	1.0e-01	3.3e-01	3.9e-01
20	1.1e-01	1.3e-01	1.1e-01	9.0e-02	3.1e-01	3.3e-01

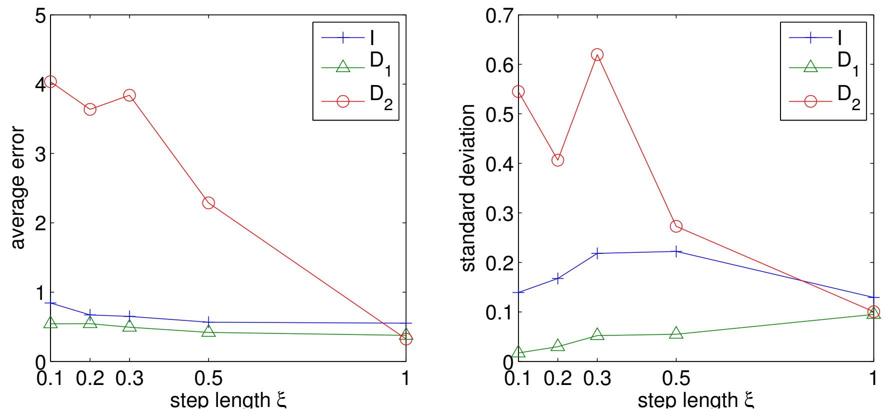


Figure 3.6: Results for the reconstruction of test function $f_{3,\xi}$ with a variable step length ξ, which is reported on the horizontal axis. The left graph reports the average error e_{opt}, obtained with three regularization matrices $M = I, D_1, D_2$. Each test is repeated 20 times for each noise level $\tau = 10^{-3}, 10^{-2}$. The right graph reports the corresponding standard deviations.

The maximum resolution which the inversion algorithm can achieve in imaging high conductivity thin layers is another important issue we inspected in this work. This situation is typical, e.g., in UXO detection. To this end, we consider the test function f_3 and let the length ξ of the step vary, that is, we set $f_{3,\xi}(z) = 1$ for $z \in [0.5, 0.5 + \xi]$ and $f_{3,\xi}(z) = 0.2$ otherwise. Each problem is solved for three regularization matrices, two noise levels, and each test is repeated 20 times for different noise realizations. The left graph of Figure 3.6 reports the average errors for different values of ξ, while the right graph displays the corresponding standard deviations. The choice $M = D_1$ appears to be the best for detecting a thin conductive layer. Indeed, not only the errors are better, but the smaller standard deviations ensure that the method is more reliable. Figure 3.7 shows the reconstructions of $f_{3,\xi}$ with three different step lengths, $\xi = 1.0, 0.5, 0.2$, $M = D_1$, and $\tau = 10^{-2}$. It is remarkable that the position of the maximum is well located by the algorithm even in the presence of a very thin step.

Remark 3.3.1. *It has been shown in recent literature [21, 42] that if a linear inverse problem is solved in L^p spaces, with $p < 2$, the reconstruction of discontinuous functions can greatly improve. This would be particularly helpful in the presence of a highly* *conductive thin layer, but applying such methods is rather involved even in the linear case.*

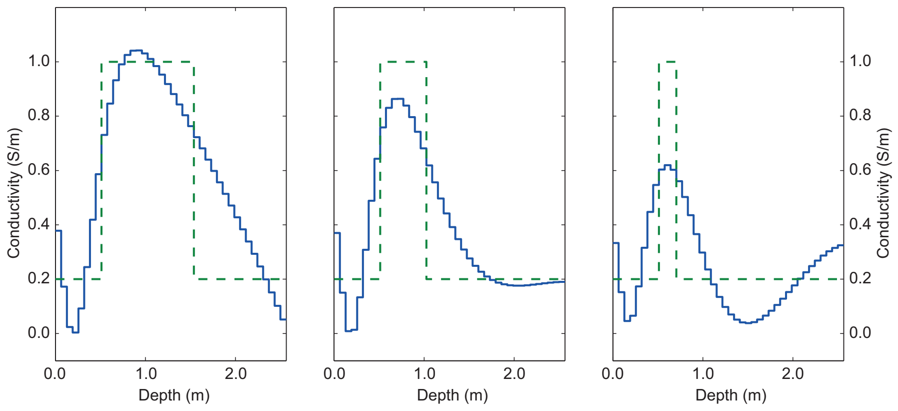


Figure 3.7: Optimal reconstructions for the test function $f_{3,\xi}$, with step lengths $\xi = 1.0$ (left), 0.5 (center), and 0.2 (right), obtained with $M = D_1$ and noise level $\tau = 10^{-2}$.

In the above experiments, the regularization parameter ℓ has been chosen optimally, that is, in order to produce the smallest deviation from the exact solution. Obviously, in real-world applications this is not possible, so it is essential to determine the parameter effectively. When an accurate estimate of the noise level is known, this can be done by the discrepancy principle (3.2.18). In our experiments, we set $\kappa = 1.5$ and substitute $\|\mathbf{e}\|$ by $\tau\|\mathbf{b}\|$, where τ is the noise level and $\mathbf{b}$ is the noisy data vector; see (3.3.1).

When the noise level is unknown, the regularization parameter may be estimated by a heuristic method. We compared the methods described in Subsection 1.6.4, namely the L-corner [72], the restricted Regińska method (ResReg) [113, 114], the residual L-curve [114, 115] and the hybrid quasi-optimality criterion [102, 114]. These methods were designed for linear inverse problems, but they can also be applied to nonlinear problems, as they only require the knowledge of the residual corresponding to each regularized solution. The L-corner method proved to be the most robust, so in the rest of the section we will only refer to it.

Our numerical experiments, showed that the discrepancy and the L-corner methods furnish very good estimates for the parameter when $M = I$, while they are less reliable when $M = D_1$ or D_2, that is, for the choice of the regularization matrix which produced the best results in our experiments. This fact is known for the L-curve; see, e.g., [116]. In fact, a good choice for M is a matrix whose kernel (approximately) contains the solution, and this makes the L-curve loose its typical "L" shape. We remark that we cannot apply the discrepancy principle to real data sets for which an estimate of the noise is not available. Moreover, according to our experience, the noise on real EMI data is not necessarily equally distributed. This will be commented on in Section 3.3.2,

Figure 3.8 shows a reconstruction of test function f_1 obtained with $m = 10$, $n = 40$, noise level $\tau = 10^{-2}$, and regularization matrix $M = D_2$. The graph on the left displays the L-curve corresponding to this example, the graph on the right compares the approximations produced by the discrepancy criterion and the L-corner method to the exact solution. In this case the optimal parameter is $\ell = 2$. The discrepancy fails, as it gives the estimate $\ell = 1$, while L-corner returns $\ell = 2$. This test function is approximately contained in the kernel of D_2, as $\|D_2\boldsymbol{\sigma}\| \simeq 3 \cdot 10^{-2}$ while $\|\boldsymbol{\sigma}\| \simeq 4$, and

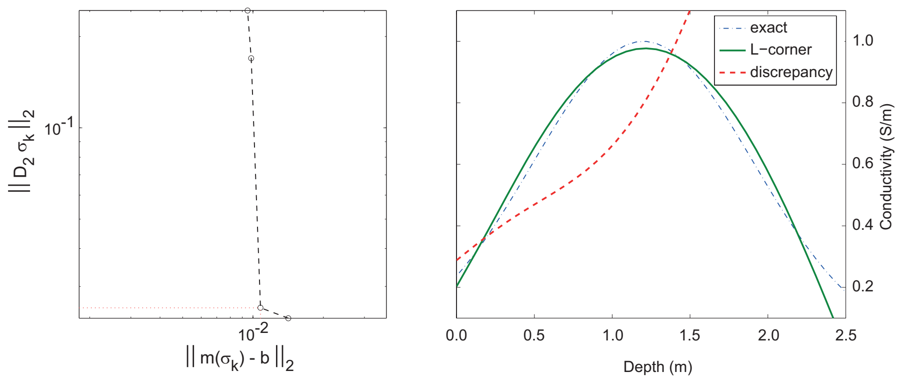


Figure 3.8: Results for test function f_1, with $m = 10$, $n = 40$, $\tau = 10^{-2}$, and $M = D_2$. The graph on the left displays the L-curve; the one on the right the exact solution and the reconstructions produced by the discrepancy principle and the L-corner method.

the L-curve appears almost shapeless. Anyway, the L-corner method implements a particular strategy to deal with such cases, and produce a good reconstruction.

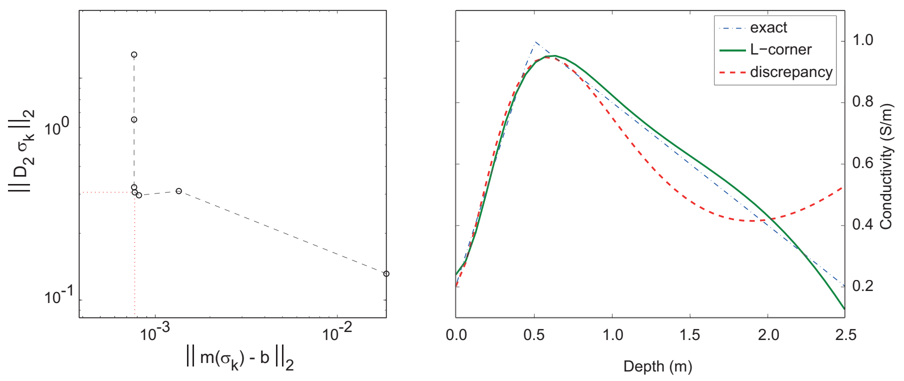


Figure 3.9: Results for test function f_2, with $m = 10$, $n = 40$, $\tau = 10^{-3}$, and $M = D_1$. The graph on the left displays the L-curve; the one on the right the exact solution and the reconstructions produced by the discrepancy principle and the L-corner method.

Figure 3.9 reports the same graphs for test function f_2, with $m = 10$, $n = 40$, $\tau = 10^{-3}$, and $M = D_1$. The optimal parameter is $\ell = 4$. The L-corner method gives $\ell = 4$ and discrepancy returns $\ell = 2$. In this case both methods succeed in identifying accurately the depth at which the conductivity is maximal.

3.3.2 Field data

We tested the nonlinear inversion technique described in the previous sections on field data collected at the Cagliari Airport (Sardinia, Italy), in an area where previous geophysical investigations, conducted for UXO detection, established the presence of layered materials with very high electrical conductivity, very suitable to be investigated with vertical electromagnetic induction soundings. The reliability of the inverted conductivity profile was assessed by comparison with conductivities obtained by electrical resistivity tomography (ERT) [32, 96].

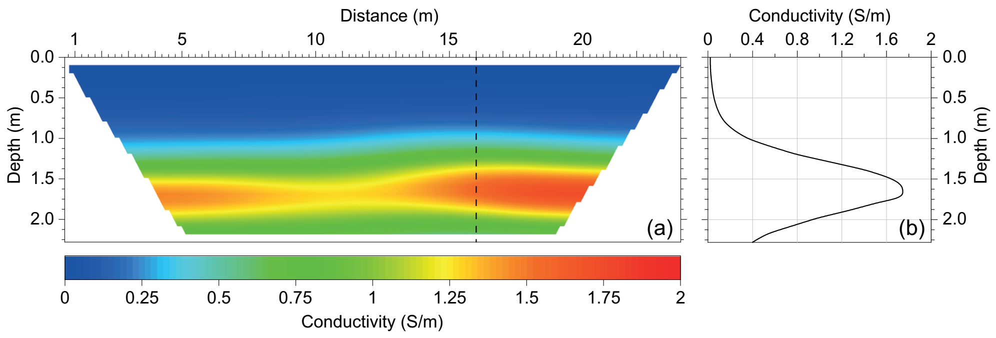
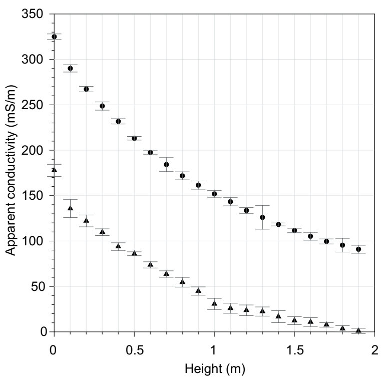


Figure 3.10: ERT results: in the left graph we display the conductivity section; the right graph reports the conductivity profile versus the depth, at the position where the electromagnetic data were collected, marked by a dashed line in the first graph.

The ERT profile was performed using 48 electrodes set up with an inter-electrode spacing of 0.5 m deployed in a Wenner-Schlumberger array, which we chose to reach a compromise between a reasonable vertical resolution and a good signal-to-noise ratio [31]. ERT data were collected using an IRIS Syscal Pro Switch 48 resistivity meter, which was set for six-cycle stacking (repetition of measurements), with the requirement of reaching a quality factor (standard deviation) of less than 5%. Data were then inverted using the commercial program Res2Dinv [94, 95]. The software employs a smoothness-constrained least-squares optimization method [29, 95] to minimize the difference between measured and modelled data, which it calculates using either a finite difference or a finite element approximation. The program divides the subsurface into rectangular cells whose corners, along the line, follow the positions of the electrodes in the subsurface [95]. The quality of the fit between measured and modelled data is expressed in terms of the root mean square (RMS) error. Figure 3.10 shows the ERT result we obtained with an RMS error of 2%, displayed in conductivity units to facilitate direct comparison with the electromagnetic data.

As expected, the section shows a subsurface model where conductivity changes almost exclusively in the vertical direction. At the near surface electrical conductivity starts with low values ($< 200\,\mathrm{mS/m}$) and keeps them down to 1 m depth; then, it abruptly increases reaching maximum values (up to $1800\,\mathrm{mS/m}$) at the depth of about 1.7 m; finally, it lowers below $800\,\mathrm{mS/m}$ in the deepest portion of the investigated section. The graph on the right of Figure 3.10 shows the conductivity profile at the location where we carried out the electromagnetic sounding. As the exact solution is not available in this real case study, this profile was used as a benchmark to comparatively assess the reliability of our regularized nonlinear inversion procedure.

Electromagnetic data were measured with the CMD-1 conductivity meter (GF Instruments), a frequency-domain electromagnetic device with a constant operating frequency of 10 kHz and 0.98 m coil separation. After completing the usual calibration procedure, the electromagnetic vertical sounding was obtained by making measurements in vertical and horizontal coil-mode configurations, lifting the instrument above the ground at heights from 0 to 1.9 m, with a 0.1 m step, by means of a specially built wooden frame; see picture on the right in Figure 3.11. For both coil orientations and



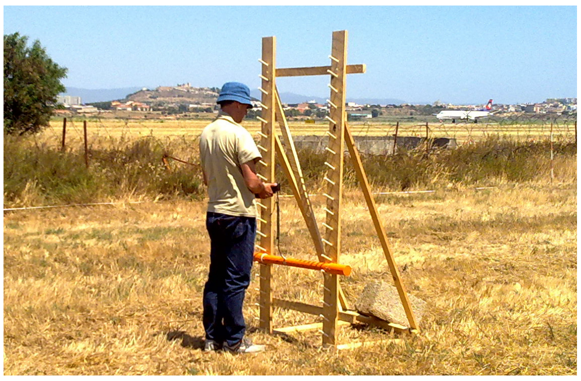


Figure 3.11: Left: mean apparent conductivities measured in vertical (circles) and horizontal (triangles) modes at different heights above the ground; error bars are standard deviations, which are multiplied by 10 for display purpose. Right: the wooden frame used to put the instrument at different heights above the ground. In the picture, the GCM is placed at height 0.5 m.

each instrument height we recorded twenty readings to get the mean value and the standard deviation of each measurement. Figure 3.11 (left) displays the resulting electromagnetic data versus height curves.

The standard deviations on the data (Figure 3.11, left) are rather different one from the other. This suggests that the noise is not equally distributed, and rules out the use of the discrepancy principle to estimate the regularization parameter, as well as other statistical methods (see, e.g., the generalized cross validation [69]) for which this assumption is essential. For this reason, in our experiments we only use heuristic parameter selection techniques.

We apply the damped Gauss–Newton method (3.2.6) to the least squares problem (3.2.3), where the vector **b** (see (3.2.2)) contains the field data reported in left graph of Figure 3.11. We use 20 measurements in vertical and horizontal orientation ($m = 20$), discretizing the soil by 40 layers ($n = 40$) up to the depth of 2.5 m. The Jacobian is exactly computed, as described in Section 3.2.2, and the damping parameter is determined by the Armijo–Goldstein principle.

Table 3.4: Performance of the methods for the estimation of the regularization parameters described in Section 3.2.4, when the inversion algorithm is applied to field data with $m = 20$, $n = 40$, and $M = I, D_1, D_2$. Each entry of the table reports the value of ℓ identified by a particular method and, in parentheses, the depth at which the maximum of $\boldsymbol{\sigma}_\ell$ is located and the value of the maximum (in S/m). The values predicted by ERT are (1.68,1.74).

	L-corner	ResReg	L-res	Q-hyb
$M = I$	5 (1.47,1.11)	4 (1.67,1.04)	2 (2.44,0.98)	7 (1.73,1.26)
$M = D_1$	2 (1.99,1.48)	1 (2.50,0.98)	2 (1.99,1.48)	2 (1.99,1.48)
$M = D_2$	2 (1.60,1.53)	1 (1.60,1.53)	2 (1.60,1.53)	1 (1.60,1.53)

We initially set $M = I$ and regularize the solution by TSVD; Figure 3.12 shows the solutions $\boldsymbol{\sigma}_\ell$, $\ell = 2, \ldots, 7$. In each graph, the dashed line represents the conductivity profile produced by ERT. To assess the performance of the L-corner method, we

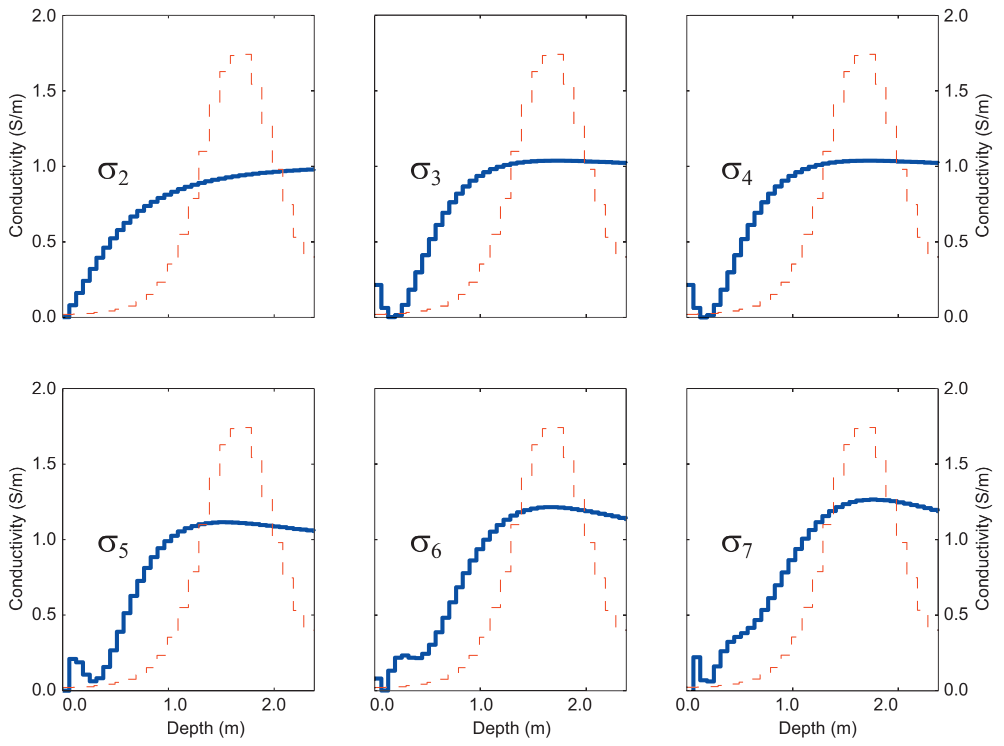
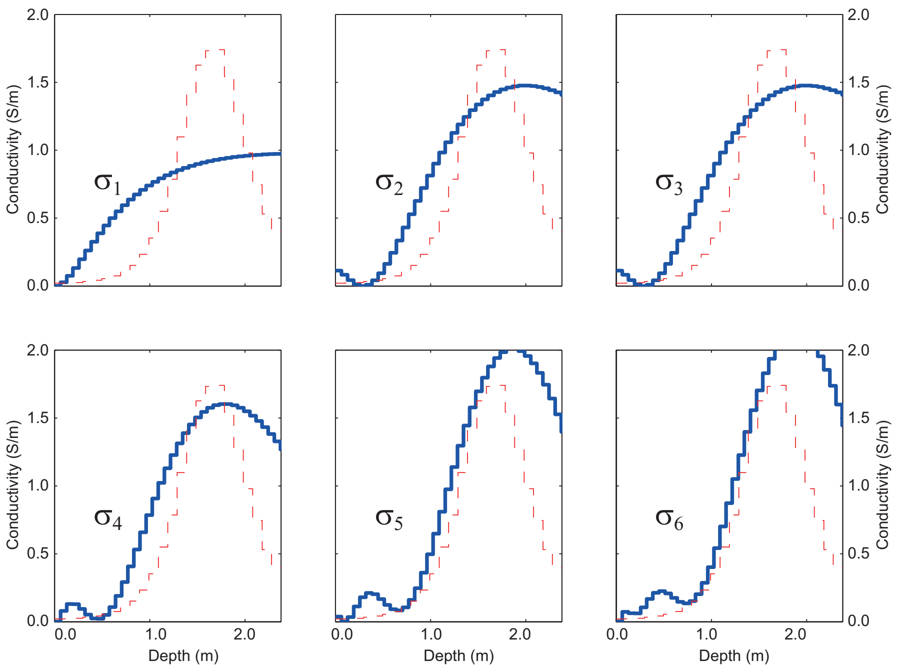


Figure 3.12: Regularized solutions $\boldsymbol{\sigma}_\ell$, with regularization parameter $\ell = 2, \ldots, 7$, obtained by applying TSVD ($M = I$) to each iteration of the Gauss–Newton method. We used all the available measurements ($m = 20$) and set $n = 40$. The dashed line represents the conductivity predicted by ERT.

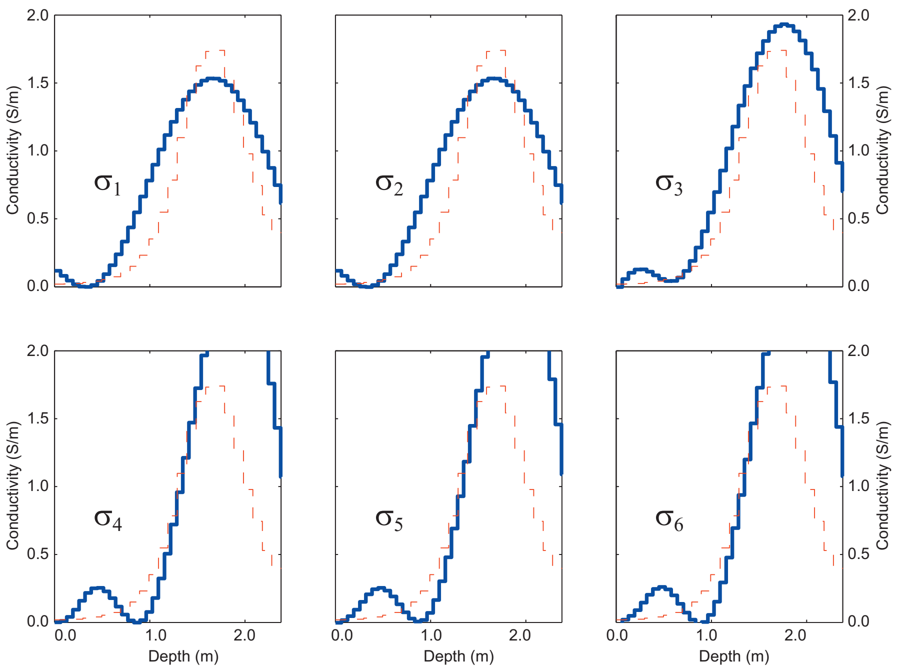
compare it to the ResReg, residual L-curve, and hybrid quasi-optimality criterions, mentioned in Section 3.3.1. The values of the regularization parameters are reported in the first row of Table 3.4, together with the coordinates (depth and value) of the maximum of the corresponding regularized solution. These are to be compared with the values (1.68,1.74) which locate the maximum of the conductivity profile predicted by ERT; see Figure 3.10, right. The results of Table 3.4 and Figure 3.12 show that the position of the maximum is well localized, starting from $\ell = 5$, but that the shape of the solution is never accurately determined.

We now report the results obtained by TGSVD with $M = D_1$ and $M = D_2$ (the discrete approximations of the first and second derivatives). The first six regularized solutions are displayed in Figure 3.13a and Figure 3.13b, respectively, together with the ERT solution. The identified regularization parameters and the coordinates of the maximal conductivity predicted by ERT appear in the second and third rows of Table 3.4.

It is clear that resorting to the TGSVD, using either the first or the second derivative as a regularizing operator, is much more effective than the TSVD approach. In this particular case, the second derivative produces the best results. Among the parameter estimation methods, the L-corner algorithm [72] appears to be the most robust, as it identifies an acceptable solution in all the three cases. Moreover, the position of the maximum is localized with sufficient accuracy.



(a) Regularized solutions $\boldsymbol{\sigma}_\ell$, with regularization parameter $\ell = 1, \ldots, 6$, obtained by applying TGSVD ($M = D_1$) to each iteration of the Gauss–Newton method. We used all the available measurements ($m = 20$) and set $n = 40$. The dashed line represents the conductivity predicted by ERT.



(b) Regularized solutions $\boldsymbol{\sigma}_\ell$, with regularization parameter $\ell = 1, \ldots, 6$, obtained by applying TGSVD ($M = D_2$) to each iteration of the Gauss–Newton method. We used all the available measurements ($m = 20$) and set $n = 40$. The dashed line represents the conductivity predicted by ERT.

Figure 3.13: Use of TGSVD

Figure 3.14 reports the first three regularized solutions with $M = D_2$, corresponding to $m = 10$ and $m = 5$, that is, using half of the measurements used in the previous figures ($h = 0\,\text{m}, 0.2\,\text{m}, \ldots, 1.8\,\text{m}$), and a quarter of them ($h = 0\,\text{m}, 0.4\,\text{m}, \ldots, 1.6\,\text{m}$). Reducing the number of data values leads to less accurate solutions, but the position of the conductivity maximum and its value are very well determined. In both cases, the L-corner method returned $\ell = 2$.

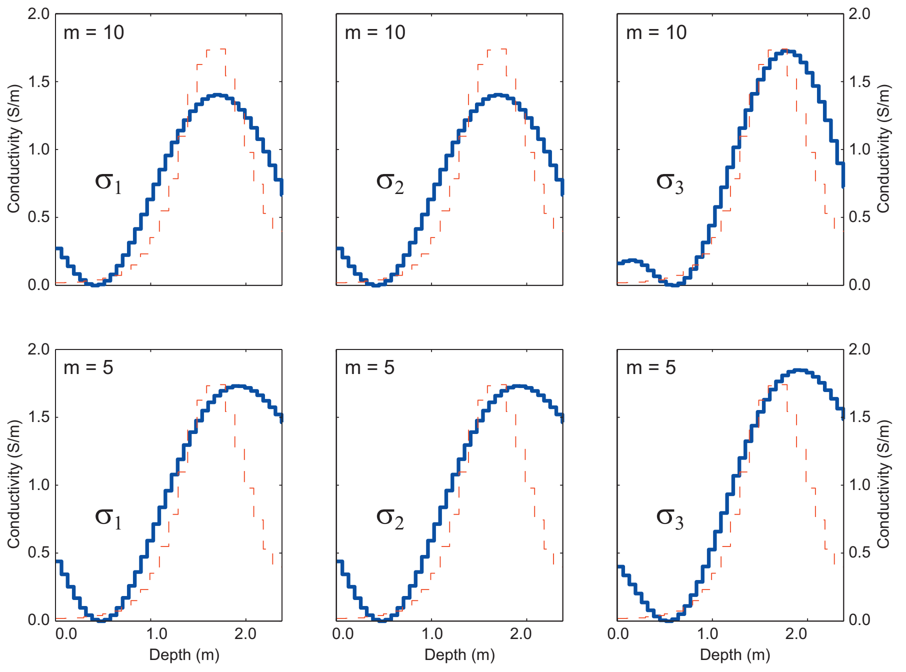


Figure 3.14: Regularized solutions $\boldsymbol{\sigma}_\ell$, with regularization parameter $\ell = 1, 2, 3$, obtained by applying TGSVD ($M = D_2$) to each iteration of the Gauss–Newton method, and setting $n = 40$. In the top graphs we used half of the available measurements ($m = 10$); the bottom graphs are obtained by $m = 5$, that is, employing a quarter of the data. The dashed line represents the conductivity predicted by ERT.

Conclusions and future work

We proposed two applications of the low-rank approximation in the fields of complex networks theory and inverse problems, respectively.

In the first one, we presented a new computational method, based on low rank approximation of the adjacency matrix, to rank the nodes of both undirected and directed networks. This method has been originally presented in [51] for undirected networks and extended for the directed ones in [5]. The numerical examples illustrate the competitiveness of the approach, called hybrid method, when applied to the analysis of large networks, and show that it produces accurate results in a reasonable time. Hybrid methods may be the only feasible approach for determining the most important nodes in terms of centrality indices expressed by bilinear forms. Moreover, we proposed to compute such indices using block methods which are more efficient on computers with a hierarchical memory structure. This analysis in presented in [50, 99]. The availability of several processors also can be utilized efficiently; see, e.g., [52] for discussions and examples.

In the second application, we proposed a regularized inversion method to reconstruct the electrical conductivity of the soil with respect to depth, starting from electromagnetic data collected by a GCM. Here the low-rank approximation is used as a regularization method since the problem is severely ill-conditioned. We develop exact formulae for the Jacobian of the function to be inverted, and choose a relaxation parameter in order to ensure both the convergence of the iterative method and the positivity of the solution. This leads to a fast and reliable algorithm. Various methods for the automatic estimation of the regularization parameter are considered. Numerical experiments on synthetic data sets show that the algorithm produces reasonable results, even when the noise level is chosen to be consistent with real applications. The method is finally applied to real field data, producing results which are compatible with those obtained by ERT.

Some topics we are working or planning to work on in the future are:

- **Complex networks theory:**
  - Recently, the study of dynamic/multi-layer networks has gained an increasing interest since they model interactions between entities which change during time. The definition of centrality indices in this case is not trivial and cannot be easily expressed using matrix functions as in the case of static networks [63, 64]. We are working on a new block matrix formulation that permits to express already introduced centrality measures in a convenient way in order to improve the computation.
  - A neuronal network is a representation of the functional connections in the brain. The mathematical model is a correlation matrix [23] which is

full and weighted. Recently, the use of graph spectral analysis has been proposed [33]. We are working on the application of some centrality indices to understand the behavior of the brain connections. Moreover, we are looking at the problem from the computational point of view.

  - Since one of the main issues in network analysis is the smart computation of centrality indices, we are working on the re-computation of these ones when a node falls, that is, when a row/coloumn of the adjacency matrix is deleted. In this case, a low-rank update has to be applied.

- **Inverse problems:**

  - We are studying an extension of the inversion of electromagnetic data to the case when not only the electrical conductivity but also the magnetic permeability varies with respect to the depth. Moreover, we plan to adapt our inversion algorithm and our software in order to deal with multiple depth responses, corresponding to multifrequency and/or multioffset GCM measurements, produced by the new generation of instruments nowadays available.

Bibliography

[1] *Alex Arena's Data Sets.* http://deim.urv.cat/~aarenas/data/welcome.htm.

[2] W. L. Anderson. "Numerical integration of related Hankel transforms of orders 0 and 1 by adaptive digital filtering". In: *Geophysics* 44.7 (1979), pp. 1287–1305.

[3] J. Baglama, D. Calvetti, and L. Reichel. "Algorithm 827: irbleigs: A MATLAB program for computing a few eigenpairs of a large sparse Hermitian matrix". In: *ACM Trans Math. Software* 29 (2003), pp. 337–348.

[4] J. Baglama, D. Calvetti, and L. Reichel. "IRBL: An implicitly restarted block Lanczos method for large-scale Hermitian eigenproblems". In: *SIAM J. Sci. Comput* 24 (2003), pp. 1650–1677.

[5] J. Baglama, C. Fenu, L. Reichel, and G. Rodriguez. "Analysis of directed networks via partial singular value decomposition and Gauss quadrature". In: *Linear Algebra and its Applications* 456 (2014), pp. 93–121.

[6] J. Baglama and L. Reichel. "An implicitly restarted block Lanczos bidiagonalization method using Leja shifts". In: *BIT* 53 (2013), pp. 285–310.

[7] J. Baglama and L. Reichel. "Augmented implicitly restarted Lanczos bidiagonalization methods". In: *SIAM J. Sci. Comput.* 27 (2005), pp. 19–42.

[8] Z. Bai, D. Day, and Q. Ye. "ABLE: An adaptive block Lanczos method for non-Hermitian eigenvalue problems". In: *SIAM J. Matrix Anal. Appl* 20 (1999), pp. 1060–1082.

[9] A. L. Barabási and R. Albert. "Emergence of scaling in random networks". In: *Science* 286.5439 (1999), pp. 509–512.

[10] M. Benzi and P. Boito. "Quadrature rule-based bounds for functions of adjacency matrices". In: *Linear Algebra Appl.* 433 (2010), pp. 637–652.

[11] M. Benzi, E. Estrada, and C. Klymko. "Ranking hubs and authorities using matrix functions". In: *Linear Algebra Appl.* 438 (2013), pp. 2447–2474.

[12] M. Benzi and C. Klymko. "Total communicability as a centrality measure". In: *J. Complex Networks* 1 (2013), pp. 124–149.

[13] D. A. Bini, G. M. Del Corso, and F. Romani. "Evaluating scientific products by means of citation-based models: a first analysis and validation". In: *Electron. Trans. Numer. Anal.* 33 (2008), pp. 1–16.

[14] *Biological Networks Data Sets of Newcastle University.* http://www.biological-networks.org/.

[15] A. Björck. *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

[16] A. Björck. *Numerical Methods for Least Squares Problems.* Philadelphia, PA: SIAM, 1996.

[17] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren. “A measure of similarity between graph vertices: applications to synonym extraction and web searching”. In: *SIAM Rev.* 46 (2004), pp. 647–666.

[18] M. Boguña, R. Pastor-Satorras, A. Dáz-Guilera, and A. Arenas. “Models of social networks based on social distance attachment”. In: *Physical Review E* 70 (2004), p. 056122.

[19] F. Bonchi, P. Esfandiar, D. F. Gleich, C. Greif, and L. V. S. Lakshmanan. “Fast matrix computations for pairwise and columnwise commute times and Katz scores”. In: *Internet Math* 8 (2012), pp. 73–112.

[20] B. Borchers, T. Uram, and J. M. H. Hendrickx. “Tikhonov regularization of electrical conductivity depth profiles in field soils”. In: *Soil Sci. Soc. Am. J.* 61.4 (1997). Package LINEM38 available at `http://infohost.nmt.edu/borchers/linem38.html`, pp. 1004–1009.

[21] P. Brianzi, F. Di Benedetto, and C. Estatico. “Preconditioned iterative regularization in Banach spaces”. In: *Comput. Optim. Appl.* 54.2 (2013), pp. 263–282.

[22] C. G. Broyden. “A class of methods for solving nonlinear simultaneous equations”. In: *Math. Comp.* 19 (1965), pp. 577–593.

[23] E. Bullmore and O. Sporns. “Complex brain networks: graph theoretical analysis of structural and functional systems”. In: *Nature Reviews Neuroscience* 10.3 (2009), pp. 186–198.

[24] D. Calvetti, G. H. Golub, and L. Reichel. “Estimation of the L-curve via Lanczos bidiagonalization”. In: *BIT* 39 (1999), pp. 603–619.

[25] D. Calvetti, L. Reichel, and F. Sgallari. “Application of anti-Gauss quadrature rules in linear algebra”. In: *Applications and Computation of Orthogonal Polynomials.* W. Gautschi and G. H. Golub and G. Opfer, eds. Birkhäuser, Basel, 1999, pp. 41–56.

[26] G. Cassiani, N. Ursino, R. Deiana, G. Vignoli, J. Boaga, M. Rossi, M. T. Perri, M. Blaschek, R. Duttmann, S. Meyer, R. Ludwig, A. Soddu, P. Dietrich, and U. Werban. “Noninvasive monitoring of soil static characteristics and dynamic states: a case study highlighting vegetation effects on agricultural land”. In: *Vadose Zone J.* 11.3 (2012).

[27] G. Concas, M. Marchesi, A. Murgia, and R. Tonelli. “An empirical study of social networks metrics in object oriented software”. In: *Advances in Software Engineering* (2010), p. 729826.

[28] G. Concas, M. Marchesi, A. Murgia, R. Tonelli, and I. Turnu. “On the distribution of bugs in the Eclipse system”. In: *IEEE Transactions on Software Engineering* 37 (2011), pp. 872–877.

[29] S. C. Constable, R. L. Parker, and C. G. Constable. “Occam’s inversion: A practical algorithm for generating smooth models from electromagnetic sounding data”. In: *Geophysics* 52.3 (1987), pp. 289–300.

[30] J. J. Croft, E. Estrada, D. J. Higham, and A. Taylor. “Mapping directed networks”. In: *Electron. Trans. Numer. Anal* 37 (2010), pp. 337–350.

[31] T. Dahlin and B. Zhou. “A numerical comparison of 2D resistivity imaging with 10 electrode arrays”. In: *Geophys. Prospect.* 52.5 (2004), pp. 379–398.

[32] W. Daily, A. Ramirez, A. Binley, and D. LeBrecque. “Electrical resistance tomography”. In: *The Leading Edge* 23.5 (2004), pp. 438–442.

[33] W. De Haan, W. M. Van Der Flier, H. Wang, P. F. A. Van Mieghem, P. Scheltens, and C. J. Stam. “Disruption of functional brain networks in Alzheimer’s disease: what can we learn from graph spectral analysis of resting-state magnetoencephalography?” In: *Brain connectivity* 2.2 (2012), pp. 45–55.

[34] Vin De Silva and Lek-Heng Lim. “Tensor rank and the ill-posedness of the best low-rank approximation problem”. In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (2008), pp. 1084–1127.

[35] G. P. Deidda, E. Bonomi, and C. Manzi. “Inversion of electrical conductivity data with Tikhonov regularization approach: some considerations”. In: *Ann. Geophys.* 46.3 (2003), pp. 549–558.

[36] G. P. Deidda, C. Fenu, and G. Rodriguez. “Regularized solution of a nonlinear problem in electromagnetic sounding”. In: *Inverse Problems* 30.12 (2014), p. 125014.

[37] J. J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Vol. 16. Philadelphia, PA: SIAM, 1983.

[38] J. Duch and A. Arenas. “Community identification using extremal optimization”. In: *Phys. Rev. E* 72 (2005), p. 027104.

[39] A. J. Duran and P. Lopez-Rodriguez. “Orthogonal matrix polynomials: zeros and Blumenthal’s theorem”. In: *J. Approx. Theory* 84 (1996), pp. 96–118.

[40] C. Eckart and G. Young. “The approximation of one matrix by another of lower rank”. In: *Psychometrika* 1.3 (1936), pp. 211–218.

[41] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems.* Dordrecht: Kluwer, 1996.

[42] C. Estatico, M. Pastorino, and A. Randazzo. “A novel microwave imaging approach based on regularization in Banach spaces”. In: *IEEE T. Antenn. Propag.* 60.7 (2012), pp. 3373–3381.

[43] E. Estrada. *The Structure of Complex Networks.* Oxford: Oxford University Press, 2012.

[44] E. Estrada and N. Hatano. “Communicability in complex networks”. In: *Phys. Rev. E* 77 (2008), p. 036111.

[45] E. Estrada, N. Hatano, and M. Benzi. “The physics of communicability in complex networks”. In: *Physics Reports* 514 (2012), pp. 89–119.

[46] E. Estrada and D. J. Higham. “Network properties revealed through matrix functions”. In: *SIAM Rev.* 52 (2010), pp. 696–714.

[47] E. Estrada, D. J. Higham, and N. Hatano. “Communicability betweenness in complex networks”. In: *Physica A* 388 (2009), pp. 764–774.

[48] E. Estrada and J. A. Rodríguez-Veláquez. “Subgraph centrality and clustering in complex hyper-networks”. In: *Physica A* 364 (2006), pp. 581–594.

[49] E. Estrada and J. A. Rodríguez-Veláquez. “Subgraph centrality in complex networks”. In: *Phys. Rev. E* 71 (2005), p. 056103.

[50] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez. “Block Gauss and anti-Gauss quadrature with application to networks”. In: *SIAM Journal on Matrix Analysis and Applications* 34.4 (2013), pp. 1655–1684.

[51] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez. “Network analysis via partial spectral factorization and Gauss quadrature”. In: *SIAM J. Sci. Comput.* 35 (2013), A2046–A2068.

[52] K. Gallivan, M. Heath, E. Ng, B. Peyton, R. Plemmons, J. Ortega, C. Romine, A. Sameh, and R. Voigt. *Parallel Algorithms for Matrix Computations*. Philadelphia: SIAM, 1990.

[53] W. Gautschi. *Orthogonal Polynomials: Computation and Approximation*. Oxford: Oxford University Press, 2004.

[54] R. Gebbers, E. Lück, and K. Heil. “Depth sounding with the EM38-detection of soil layering by inversion of apparent electrical conductivity measurements”. In: *Precision Agriculture ’07*. Ed. by J. V. Stafford. The Netherlands: Wageningen Academic Publisher, 2007, pp. 95–102.

[55] *Gephi Sample Data Sets*. http://wiki.gephi.org/index.php/Datasets.

[56] G. H. Golub. “Bounds for matrix moments”. In: *Rocky Moutain J. Math* 4 (1974), pp. 207–211.

[57] G. H. Golub, M. Heath, and G. Wahba. “Generalized cross-validation as a method for choosing a good ridge parameter”. In: *Technometrics* 21.2 (1979), pp. 215–223.

[58] G. H. Golub and G. Meurant. “Matrices, moments and quadrature”. In: *Numerical Analysis 1993* 303 (1994), pp. 105–156.

[59] G. H. Golub and G. Meurant. *Matrices, Moments and Quadrature with Applications*. Princeton: Princeton University Press, 2010.

[60] G. H. Golub and C. F. Van Loan. *Matrix Computations*. third. Baltimore: The John Hopkins University Press, 1996.

[61] L. Grasedyck, D. Kressner, and C. Tobler. “A literature survey of low-rank tensor approximation techniques”. In: *GAMM-Mitteilungen* 36.1 (2013), pp. 53–78.

[62] J. P. Greenhouse and D. D. Slaine. “The use of reconnaissance electromagnetic methods to map contaminant migration”. In: *Ground Water Monit. Remediat.* 3.2 (1983), pp. 47–59.

[63] P. Grindrod and D. J. Higham. “A matrix iteration for dynamic network summaries”. In: *SIAM Review* 55.1 (2013), pp. 118–128.

[64] P. Grindrod, M. C. Parsons, D. J. Higham, and E. Estrada. “Communicability across evolving networks”. In: *Physical Review E* 83.4 (2011), p. 046120.

[65] R. Guimerà, L. Danon, A. Dí az-Guilera, F. Giralt, and A. Arenas. “Self-similar community structure in a network of human interactions”. In: *Phys. Rev. E* 68 (2003), 065103(R).

[66] Jacques Hadamard. *Lectures on Cauchy’s problem in linear partial differential equations*. New Haven: Yale University Press, 1923.

[67] M. Hanke. *Conjugate gradient type methods for ill-posed problems*. Vol. 327. Longman, Harlow, UK: Pitman Research Notes in Mathematics, 1995.

[68] M. Hanke. “Limitations of the L-curve method in ill-posed problems”. In: *BIT* 36 (1996), pp. 287–301.

[69] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems, Numerical Aspects of Linear Inversion*. Philadelphia, PA: SIAM, 1998.

[70] P. C. Hansen. “Regularization Tools: Version 4.0 for Matlab 7.3”. In: *Numer. Algorithms* 46 (2007), pp. 189–194.

[71] P. C. Hansen. “The truncated SVD as a method for regularization”. In: *BIT* 27 (1987), pp. 543–553.

[72] P. C. Hansen, T. K. Jensen, and G. Rodriguez. “An adaptive pruning algorithm for the discrete L-curve criterion”. In: *J. Comput. Appl. Math.* 198.2 (2007), pp. 483–492.

[73] P. C. Hansen and D. P. O’Leary. “The use of the L-curve in the regularization of discrete ill-posed problems”. In: *SIAM J. Sci. Comput.* 14 (1993), pp. 1487–1503.

[74] J.M.H. Hendrickx, B. Borchers, D.L. Corwin, S.M. Lesch, A.C. Hilgendorf, and J. Schlue. “Inversion of soil conductivity profiles from electromagnetic induction measurements”. In: *Soil Sci. Soc. Am. J.* 66.3 (2002). Package NONLINEM38 available at http://infohost.nmt.edu/~borchers/nonlinem38.html, pp. 673–685.

[75] V. E. Henson and G. Sanders. “Locally supported eigenvectors and matrices associated with connected and unweighted power-law graphs, Electron”. In: *Electron. Trans. Numer. Anal* 39 (2012), pp. 353–378.

[76] M. R. Hestenes and E. Stiefel. “Methods of Conjugate Gradients for Solving Linear Systems1”. In: *Journal of Research of the National Bureau of Standards* 49.6 (1952).

[77] N. J. Higham. *Functions of Matrices: Theory and Computation*. Philadelphia: SIAM, 2008.

[78] M. E. Hochstenbach, L. Reichel, and G. Rodriguez. “Regularization parameter determination for discrete ill-posed problems”. In: *J. Comput. Appl. Math.* 273 (2015), pp. 132–149.

[79] H. Huang, B. SanFilipo, A. Oren, and I. J. Won. “Coaxial coil towed EMI sensor array for UXO detection and characterization”. In: *J. Appl. Geophys.* 61.3 (2007), pp. 217–226.

[80] H. Huang and I. J. Won. “Automated anomaly picking from broadband electromagnetic data in an unexploded ordnance (UXO) survey”. In: *Geophysics* 68.6 (2003), pp. 1870–1876.

[81] H. Jeong, S. Mason, A.-l. Barabási, and Z. N. Oltvai. “Lethality and centrality of protein networks”. In: *Nature* 411 (2001), pp. 41–42.

[82] J. Kleinberg. “Authorative sources in hyperlinked environments”. In: *J. ACM* 49 (1999), pp. 604–632.

[83] T. G. Kolda and B. W. Bader. “Tensor decompositions and applications”. In: *SIAM review* 51.3 (2009), pp. 455–500.

[84] G. López Lagomasino, L. Reichel, and L. Wunderlich. “Matrices, moments, and rational quadrature”. In: *Linear Algebra Appl.* 429 (2008), pp. 2540–2554.

[85] S. Lang. *Linear algebra*. Springer, 1992.

[86] E. Lascano, P. Martinelli, and A. Osella. “EMI data from an archaeological resistive target revisited”. In: *Near Surf. Geophys.* 4.6 (2006), pp. 395–400.

[87] D. P. Laurie. “Anti-Gaussian quadrature formulas”. In: *Math. Comp.* 65 (1996), pp. 739–747.

[88] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *Arpack Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* Vol. 6. SIAM Philadelphia, 1998.

[89] S. M. Lesch, D. J. Strauss, and J. D. Rhoades. “Spatial prediction of soil salinity using electromagnetic induction techniques: 1. Statistical prediction models: A comparison of multiple linear regression and cokriging”. In: *Water Resour. Res.* 31.2 (1995), pp. 373–386.

[90] J. Leskovec, D. Huttenlocher, and J. Kleinberg. “Predicting positive and negative links in online social networks”. In: *Proceedings of the 19th international conference on World Wide Web (WWW '10).* 2010, pp. 641–650.

[91] J. Leskovec, D. Huttenlocher, and J. Kleinberg. “Signed networks in social media”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10).* 2010, pp. 1361–1370.

[92] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. “Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters”. In: *Internet Math* 6 (2009), pp. 29–123.

[93] *Lev Muchnik's Data Sets web page.* `http://www.levmuchnik.net/Content/Networks/NetworkData.html`.

[94] M. H. Loke, I. Acworth, and T. Dahlin. “A comparison of smooth and blocky inversion methods in 2D electrical imaging surveys”. In: *Explor. Geophys.* 34.3 (2003), pp. 182–187.

[95] M. H. Loke and R. D. Barker. “Practical techniques for 3D resistivity surveys and data inversion”. In: *Geophys. Prospect.* 44.3 (1996), pp. 499–523.

[96] M. H. Loke and R. D. Barker. “Rapid least-squares inversion of apparent resistivity pseudosections by a quasi-Newton method”. In: *Geophys. Prospect.* 44.1 (1996), pp. 131–152.

[97] J. Marcelino and M. Kaiser. “Critical paths in a metapopulation model of H1N1: Efficiently delaying influenza spreading through flight cancellation”. In: *PLoS Curr.* 23 (Apr. 2012), e4f8c9a2e1fca8.

[98] *Mark Newman's web page.* `http://www-personal.umich.edu/~mejn/netdata/`.

[99] D. R. Martin. “Quadrature approximation of matrix functions, with applications”. PhD Thesis. Kent State University, 2012.

[100] P. Martinelli and M. C. Duplaá. “Laterally filtered 1D inversions of small-loop, frequency-domain EMI data from a chemical waste site”. In: *Geophysics* 73.4 (2008), F143–F149.

[101] J. D. McNeill. *Electromagnetic terrain conductivity measurement at low induction numbers.* Tech. rep. TN-6. Mississauga, Ontario, Canada: Geonics Limited, 1980.

[102] V. A. Morozov. *Methods for Solving Incorrectly Posed Problems.* New York: Springer Verlag, 1984.

[103] M. E. J. Newman. *Networks: An Introduction.* Oxford: Oxford University Press, 2010.

[104] M. E. J. Newman. “The structure of scientific collaboration networks”. In: *Proc. Natl. Acad. Sci. USA* 98 (2001), pp. 404–409.

[105] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic Press, 1970.

[106] A. Osella, M. de la Vega, and E. Lascano. “3D electrical imaging of an archaeological site using electrical and electromagnetic methods”. In: *Geophysics* 70.4 (2005), G101–G107.

[107] C. C. Paige and M. A. Saunders. “Towards a generalized singular value decomposition”. In: *SIAM J. Numer. Anal.* 18.3 (1981), pp. 398–405.

[108] J. G. Paine. “Determining salinization extent, identifying salinity sources, and estimating chloride mass using surface, borehole, and airborne electromagnetic induction methods”. In: *Water Resour. Res.* 39.3 (2003).

[109] *Pajek Data Sets*. `http://vlado.fmf.uni-lj.si/pub/networks/data/`.

[110] L. Pellerin. “Applications of electrical and electromagnetic methods for environmental and geotechnical investigations”. In: *Surv. Geophys.* 23.2-3 (2002), pp. 101–132.

[111] M. Penrose. *Geometric Random Graphs*. Oxford: Oxford University Press, 2003.

[112] H. Ramsin and P. Å. Wedin. “A comparison of some algorithms for the nonlinear least squares problem”. In: *BIT Numerical Mathematics* 17.1 (1977), pp. 72–90.

[113] T. Regińska. “A regularization parameter in discrete ill-posed problems”. In: *SIAM J. Sci. Comput.* 17 (1996), pp. 740–749.

[114] L. Reichel and G. Rodriguez. “Old and new parameter choice rules for discrete ill-posed problems”. In: *Numer. Algorithms* 63.1 (2013), pp. 65–87.

[115] L. Reichel and H. Sadok. “A new L-curve for ill-posed problems”. In: *J. Comput. Appl. Math.* 219 (2008), pp. 493–508.

[116] G. Rodriguez and D. Theis. “An algorithm for estimating the optimal regularization parameter by the L-curve”. In: *Rend. Mat. Appl.* 25 (2005), pp. 69–84.

[117] L. Sambuelli, S. Leggieri, C. Calzoni, and C. Porporato. “Study of riverine deposits using electromagnetic methods at a low induction number”. In: *Geophysics* 72.5 (2007), B113–B120.

[118] G. Schultz and C. Ruppel. “Inversion of inductive electromagnetic data in highly conductive terrains”. In: *Geophysics* 70.1 (2005), G16–G28.

[119] A. Sinap and W. Van Assche. “Polynomial interpolation and Gaussian quadrature for matrix-valued functions”. In: *Linear Algebra Appl.* 207 (1994), pp. 71–114.

[120] *Snap Network Data Sets*. `http://snap.stanford.edu/data/index.html`.

[121] B. R. Spies and F. C. Frischknecht. “Electromagnetic sounding”. In: *Electromagnetic Methods in Applied Geophysics. Volume 2: Application*. Ed. by M. N. Nabighian. Vol. 3. Investigation in Geophysics. Tulsa, OK: Society of Exploration Geophysicists, 1991. Chap. 5, pp. 285–426.

[122] S. Sun, L. Ling, N. Zhang, G. Li, and R. Chen. “Topological structure analysis of the protein-protein interaction network in budding yeast”. In: *Nucleic Acids Research* 31 (2003), pp. 2443–2450.

[123] A. Taylor and D. J. Higham. "CONTEST: A controllable test matrix toolbox for MATLAB". In: *ACM Trans. Math. Software* 35 (2009), pp. 1–26.

[124] *The Max Plank Institute for Software Systems web site.* http://socialnetworks.mpi-sws.org/data-wosn2009.html.

[125] *The University of Florida Sparse Matrix Collection.* http://www.cise.ufl.edu/research/sparse/matrices/.

[126] J. Thiesson, M. Dabas, and S. Flageul. "Detection of resistive features using towed Slingram electromagnetic induction instruments". In: *Archaeol. Prospect.* 16.2 (2009), pp. 103–109.

[127] L. N. Trefethen and D. Bau III. *Numerical linear algebra.* Vol. 50. Siam, 1997.

[128] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. "On the evolution of user interaction in Facebook". In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09).* Ed. by Spain Barcelona. 2009, pp. 37–42.

[129] C. R. Vogel. "Non-convergence of the L-curve regularization parameter selection method". In: *Inverse Problems* 12.4 (1996), p. 535.

[130] G. Wahba. "Practical approximate solutions to linear operator equations when the data are noisy". In: *SIAM Journal on Numerical Analysis* 14.4 (1977), pp. 651–667.

[131] J. R. Wait. *Geo-Electromagnetism.* New York: Academic Press, 1982.

[132] S. H. Ward and G. W. Hohmann. "Electromagnetic theory for geophysical applications". In: *Electromagnetic Methods in Applied Geophysics. Volume 1: Theory.* Ed. by M. N. Nabighian. Vol. 3. Investigation in Geophysics. Tulsa, OK: Society of Exploration Geophysicists, 1987. Chap. 4, pp. 131–311.

[133] D. J. Watts and S. H. Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393 (1998), pp. 440–442.

[134] R. Yao and J. Yang. "Quantitative evaluation of soil salinity and its spatial distribution using electromagnetic induction method". In: *Agric. Water Manage.* 97.12 (2010), pp. 1961–1970.

[135] T. Zhang and G. H. Golub. "Rank-one approximation to high order tensors". In: *SIAM Journal on Matrix Analysis and Applications* 23.2 (2001), pp. 534–550.