Direct and iterative methods for large sparse linear systems

Nikoletta Chatzichampi

UNIVERSITY OF CRETE

Department of Mathematics and Applied Mathematics

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
Τμήμα Μαθηματικών και Εφαρμοσμένων Μαθηματικών

Direct and iterative methods for large sparse linear systems

Nikoletta Chatzichampi

Διπλωματική εργασία υποβληθείσα προς μερική εκπλήρωση
των απαραίτητων προϋποθέσεων για την απόκτηση του
Μεταπτυχιακού Διπλώματος Ειδίκευσης
στα Εφαρμοσμένα Μαθηματικά

4 Ιουλίου 2023

Επιβλέπων:

Μιχάλης Πλεξουσάκης

Μέλη επιτροπής:

Μιχάλης Πλεξουσάκης

Θεόδωρος Κατσαούνης

Παναγιώτης Χατζηπαντελίδης

Ημερομηνία εξέτασης: 4 Ιουλίου 2023

Acknowledgements

During my undergraduate studies at the Department of Mathematics of Aristotle University of Thessaloniki (AUTH) and my graduate studies at Department of Mathematics and Applied Mathematics of University of Crete, I noticed that my scientific interests lie in the field of numerical analysis and for this reason I chose a related subject. With the guidance of my supervisor, I decided to focus on iterative methods for large sparse linear systems which compose an integral part of solving problems arise from real applications.

I would like to express my deepest gratitude to my supervisor Michael Plexousakis for his guidance and patience in supervising this thesis. I have been extremely lucky to have a supervisor who cared so much about my work. Except for the knowledge that I have gained, I dealt with troubles which taught me that every difficulty can be overcome. It was very important for me that I knew I could ask for his help through all the process of writing this thesis.

In addition, I would like to thank my friend and classmate in AUTH who taught me, through our conversations, that if we believe in ourselves we can always achieve our goals. Unfortunately, he passed away a couple of months ago so, i would like to dedicate this thesis in memory of him.

Lastly, I would be remiss if I did not thank my family and specially my parents for their support and their belief in me through these years. Without their love and support I wouldn't be the person I am today. They provided me with the most important skills which are the values that they taught me. I owe all my achievements to them.

Contents

Acknowledgements **v**

List of figures **ix**

List of tables **xi**

Περίληψη **xii**

Abstract **xiii**

1 Introduction **1**

2 Direct methods for linear systems **3**
2.1 Finite Difference Methods . . . 3
2.1.1 An upwind scheme . . . 7
2.1.2 Cyclic Reduction . . . 9
2.2 The coordinate format . . . 10
2.3 Solving sparse linear system using Python . . . 12
2.3.1 Package Numpy . . . 12
2.3.2 Sparse matrices (scipy.sparse) . . . 12
2.3.3 Sparse Linear Algebra (scipy.sparse.linalg) . . . 13
2.3.4 Applications and examples . . . 13

3 Stationary Iterative methods **17**
3.1 Stationary Iterative Methods . . . 17
3.1.1 A short description of Jacobi and Gauss-Seidel Methods . . . 17
3.1.2 Matrix splittings for the Jacobi and the Gauss–Seidel methods . . . 18
3.1.3 Convergence of iterative methods . . . 19
3.2 Linear iterative schemes . . . 24
3.3 Matrix splitting methods . . . 28

4 Gradient methods and Krylov subspaces **33**
4.1 Introductory concepts . . . 33
4.2 Gradient descent methods . . . 38
4.3 The steepest descent method . . . 39

4.4 The conjugate gradient method 43
4.4.1 Non-zero starting values 56
4.4.2 Preconditioned conjugate gradient method 57
4.4.3 Krylov subspace methods for non HPD problems 58

5 The Helmholtz Equation **61**
5.1 The Helmholtz equation 61
5.2 A finite difference method for the model problem 62
5.3 A finite element method for the model problem 67

References **71**

Appendices **73**

A First appendix **73**

List of Figures

2.1 Finite difference solution and exact solution of the convection-diffusion equation (2.14) with $a = 2$, $b = 9$ and $N = 30$ . . . 8
2.2 A comparison between the upwind scheme choice and the forward approximation, in the case of $b > 0$ . . . 8
2.3 Graph of bcsstk16 . . . 13
2.4 Graph of bcsstk17 . . . 13
2.5 Graph of bcsstk25 . . . 14
2.6 Graph of bcsstk36 . . . 14
2.7 Graph of wathen100 . . . 14
2.8 Results of run code A.2 . . . 15
2.9 Comparison plot between time (also normalized time) and non-zero elements . . . 15

3.1 Results for GS and SOR methods . . . 31
3.2 Comparison plot between residual error and iteration . . . 31

5.1 Graph of the domain $\Omega = (0,1) \times (0,1)$ and the corresponding outward unit normal vectors. 64
5.2 Schematic represenatation of the waveguide. . . . 67
5.3 The ILU(0) factorization for a five–point matrix as it shown in [6] . . . 68

List of Tables

4.1 Results for bcsstk01 49
4.2 Results for bcsstk16 49

5.1 Comparison of GMRES, BiCGStab and SuperLU 69

Περίληψη

Η παρούσα εργασία έχει ως κύριο στόχο την παρουσίαση *άμεσων και επαναληπτικών μεθόδων* για την επίλυση μεγάλων και αραιών γραμμικών συστημάτων της μορφής $Ax = b$ όπου $b \in \mathbb{C}^n$ και $A \in \mathbb{C}^{n\times n}$.

Στον τομέα της αριθμητικής ανάλυσης όσο και στους επιστημονικούς υπολογισμούς, είναι σημαντικό να χρησιμοποιούμε όσο το δυνατόν λιγότερη μνήμη για την αποθήκευση δεδομένων ενός προβλήματος. Τα αραιά συστήματα μας δίνουν την δυνατότητα εξοικονόμησης μνήμης αποθηκεύοντας μόνο τα μη μηδενικά δεδομένα σε ειδικές μορφές, οι οποίες παρουσιάζονται στο δεύτερο κεφάλαιο. Ένα ακόμη πλεονέκτημα των αραιών συστημάτων είναι το γεγονός ότι μπορούμε να μειώσουμε το υπολογιστικό κόστος αφού γνωρίζουμε εκ των προτέρων το αποτέλεσμα των πράξεων με μηδενικά στοιχεία. Είναι σημαντικό να μπορούμε να εξισορροπήσουμε τα τρία αυτά βασικά χαρακτηριστικά, δηλαδή τον χώρο αποθήκευσης, το υπολογιστικό κόστος και την ευστάθεια της υπολογιστικής διαδικασίας ούτως ώστε να οδηγηθούμε σε μια αποτελεσματική λύση του προβλήματος. Στο Κεφάλαιο 2 παρουσιάζονται, επίσης, παραδείγματα προβλημάτων αρχικών τιμών τα οποία διακριτοποιούνται με τη *μέθοδο των πεπερασμένων διαφορών* και οδηγούν σε αραιά γραμμικά συστήματα εξισώσεων.

Ακολούθως, στο Κεφάλαιο 3 παρουσιάζονται μέθοδοι της κατηγορίας *Stationary Iterative Methods* στην οποία περιλαμβάνονται οι μέθοδοι *Jacobi*, *Gauss–Seidel*, καθώς επίσης και η *μέθοδος της διαδοχικής υπερχαλάρωσης* (SOR). Δίνουμε έμφαση τόσο στην περιγραφή αυτών των επαναληπτικών σχημάτων, όσο και στην ανάλυση των ιδιοτήτων σύγκλισης.

Στο Κεφάλαιο 4 παρουσιάζονται οι υπόχωροι Krylov και οι μέθοδοι *απότομης καθόδου* και *συζυγών κλίσεων*. Τέλος, στο Κεφάλαιο 5 εξετάζουμε την αποτελεσματικότητα και τη σχετική απόδοση κατάλληλων άμεσων και επαναληπτικών μεθόδων σε συστήματα γραμμικών εξισώσεων που προκύπτουν από τη διακριτοποίηση της εξίσωσης του Helmholtz με μεθόδους πεπερασμένων διαφορών και πεπερασμένων στοιχείων.

Abstract

In this thesis presented an overview of direct and iterative methods for solving large sparse linear systems such as $Ax = b$ where $A \in \mathbb{C}^{n\times n}$ and $b \in \mathbb{C}^n$.

In numerical analysis and scientific computing, an important condition for the computations is the low consuming of memory storage. A sparse linear system has the advantage that the amount of storage required is greatly reduced and several storage schemes have been devised for this special category. The respective theory was developed in the second chapter. In addition, the computational cost is reduced since we know beforehand the result of arithmetic operations with zero. The main challenge in sparse linear algebra is to balance storage, computational cost and stability to create an effective solution. In chapter 2, it is also described the *Finite Difference Methods*, a class of numerical techniques for solving differential equations by approximating derivatives with finite differences.

Later in chapter 3, there are developed the *Stationary Iterative Methods*. Some of them are the well–known methods of Jacobi, Gauss–Seidel and SOR method as well. Consequently, we emphasize the *linear iterative schemes* that constitute an important part of iterative methods.

Continuing into Chapter 4, we define the *Krylov subspace*. The jth Krylov subspace formed by the linear combination of $b, Ab, \cdots, A^{j-i}b$ and comprises the base of *Gradient methods*. We will refer to the *steepest descent method* and the *conjugate gradient method* which differ in the search direction.

Finally, in Chapter 5 we present the results of numerical experiments performed with the solution of linear systems which result from the discretization of the *Helmholtz* equation, using finite difference and finite element methods. The Python codes used in the numerical experiments performed in this thesis are listed in Appendix A.

CHAPTER 1

Introduction

When discretizing partial differential equations, we often end up with a system of linear equations in the form of $Ax = b$, where A is a, generally, sparse matrix, and b is a known vector. In the context of partial differential equations, A typically arises from a finite difference, finite element, or finite volume discretization of the differential operators in the equation.

Sparse linear systems are particularly challenging to solve because direct methods, like Gaussian elimination of the Cholesky factorization can be very inefficient due to the large number of zero entries. Therefore, iterative methods such as the Conjugate Gradient method or the Generalized Minimal Residual method are often used instead. These methods exploit the sparsity of the matrix to reduce the computational complexity of the solution process.

Many specialized algorithms and libraries have been developed specifically for solving sparse linear systems arising from partial differential equations, such as the Sparse Solvers Library (SPLIB) and the SuiteSparse package. These libraries use a variety of techniques to accelerate the solution process, such as reordering the matrix to reduce fill-in and using parallel processing.

CHAPTER 2

Direct methods for linear systems

2.1 Finite Difference Methods

Suppose that we have the function $u \in C^4$. Using Taylor's formula we have

$$u(x+h) = u(x) + h\frac{du}{dx} + \frac{h^2}{2}\frac{d^2u}{dx^2} + \frac{h^3}{6}\frac{d^3u}{dx^3} + \frac{h^4}{24}\frac{d^4u}{dx^4}(\xi_+) \tag{2.1}$$

for some $\xi_+ \in (x, x+h)$. Therefore, from (2.1) we conclude that:

$$\frac{du}{dx} = \frac{u(x+h)-u(x)}{h} + \frac{h}{2}\frac{d^2u(x)}{dx^2} + O(h^2). \tag{2.2}$$

In a similar fashion,

$$u(x-h) = u(x) - h\frac{du}{dx} + \frac{h^2}{2}\frac{d^2u}{dx^2} - \frac{h^3}{6}\frac{d^3u}{dx^3} + \frac{h^4}{24}\frac{d^4u}{dx^4}(\xi_-), \tag{2.3}$$

for some $\xi_- \in (x-h, x)$. Using (2.1) and (2.3) we obtain the approximation of the second derivative

$$\frac{d^2u}{dx^2} = \frac{u(x+h)-2u(x)+u(x-h)}{h^2} + \frac{h^2}{12}\frac{d^4u}{dx^4}(\xi) \tag{2.4}$$

The approximation (2.4) is called a *centered difference approximation* of the second derivative while (2.1) is a so-called *forward approximation* of the first derivative and (2.3) is a *backward approximation*. The combination of (2.1) and (2.3) gives the centered difference formula

$$\frac{du}{dx} \approx \frac{u(x+h)-u(x-h)}{2h}. \tag{2.5}$$

For future reference we define the forward and backward difference operators by

$$\delta^+ u(x) = u(x+h) - u(x), \tag{2.6}$$
$$\delta^- u(x) = u(x) - u(x-h). \tag{2.7}$$

Consider now the boundary value problem

$$-u''(x) = f(x), \quad x \in (0,1), \tag{2.8}$$
$$u(0) = u(1) = 0, \tag{2.9}$$

and a discretization of the interval $[0,1]$

$$x_i = ih, \quad i = 0,\dots,n+1,$$

where $h = \frac{1}{n+1}$ for some integer $n \geq 2$. Motivated by (2.4), we define approximations U_i of the values of $u(x_i)$, $i = 0,\dots,n+1$, by

$$-U_{i-1} + 2U_i - U_{i+1} = h^2 f_i, \qquad i = 1,\dots,n, \tag{2.10}$$

with $U_0 = U_{n+1} = 0$ and $f_i = f(x_i)$, $i = 1,\dots,n$. The equations in (2.10) may be written as

$$AU = F,$$

where $U = (U_1,\dots,U_n)^T$, $F = h^2(f(x_1),\dots,f(x_n))^T$, and

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}. \tag{2.11}$$

The tridiagonal matrix A is symmetric positive definite so a solution of the linear system represented by (2.10) may be easily computed via the *Cholesky factorization* which we present below:

Theorem 2.1 (Cholesky factorization)**.** *Given a symmetric positive definite matrix A, there exists a lower triangular matrix L such that*

$$A = LL^T.$$

Proof. We proceed by induction on the dimension of the matrix A. Indeed, for a 1×1 positive definite matrix $A = (\alpha_{11})$, the assertion holds with $L_{11} = \sqrt{\alpha_{11}}$ (recall that the diagonal elements of a positive definite matrix are positve). Assume now that it holds for all $(n-1) \times (n-1)$ positive definite matrices. For $A \in \mathbb{R}^{n,n}$ symmetric positive definite, we write

$$A = \begin{pmatrix} d & u^T \\ u & \tilde{H} \end{pmatrix},$$

where $d = \alpha_{11} > 0$, u is a $(n-1)-$column vector and $\tilde{H} \in \mathbb{R}^{n-1,n-1}$. We observe that

$$A = \begin{pmatrix} d & 0 \\ \frac{u}{\sqrt{d}} & I_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} \sqrt{d} & \frac{u^T}{\sqrt{d}} \\ 0 & I_{n-1} \end{pmatrix}, \tag{2.12}$$

where

$$H = \tilde{H} - \frac{1}{d} uu^T.$$

The matrix H is symmetric positive define matrix since for $x \in \mathbb{R}^{n-1}, x \neq 0$ and

$$y = \begin{pmatrix} -\frac{1}{d} x^T u \\ x \end{pmatrix} \in \mathbb{R}^n,$$

we have

$$x^T H x = x^T \left(\tilde{H} - \frac{1}{d} uu^T \right) x = y^T \begin{pmatrix} d & u^T \\ u & \tilde{H} \end{pmatrix} y = y^T A y > 0,$$

and clearly $y \neq 0$. From the inductive hypothesis, H can be written as $H = L_H L_H^T$, where L_H a lower triangular matrix with positive diagonal elements. Therefore, as a result of (2.12), A is equal to

$$A = \begin{pmatrix} \sqrt{d} & 0 \\ \frac{u}{\sqrt{d}} & I_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & L_H \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & L_H^T \end{pmatrix} \begin{pmatrix} \sqrt{d} & \frac{u^T}{\sqrt{d}} \\ 0 & I_{n-1} \end{pmatrix} = \begin{pmatrix} \sqrt{d} & 0 \\ \frac{u}{\sqrt{d}} & L_H \end{pmatrix} \begin{pmatrix} \sqrt{d} & \frac{u^T}{\sqrt{d}} \\ 0 & L_H^T \end{pmatrix},$$

so, for

$$L := \begin{pmatrix} \sqrt{d} & 0 \\ \frac{u}{\sqrt{d}} & L_H \end{pmatrix},$$

we have the required factorization $A = LL^T$. The proof of the uniqueness of L is as follows:
Assume that there exists a lower triangular matrix M with positive diagonal elements such that $A = LL^T = MM^T$. In this case $L^{-1}M = L^T \left(M^T\right)^{-1}$. On the left side of equality we have a lower triangular matrix and on the right side we have an upper triangular matrix so $L^{-1}M = L^T \left(M^T\right)^{-1} = D$, where D is a diagonal matrix. Thus,

$$M = LD \Rightarrow D_{ii} = M_{ii}/L_{ii},$$

and

$$L^T = DM^T \Rightarrow D_{ii} = L_{ii}/M_{ii}.$$

Therefore, $L_{ii}^2 = M_{ii}^2 \Rightarrow L_{ii} = M_{ii}$ and $D = I_n$, which implies that $M = L$. □

The elements of the matrix L may be easily computed by comparing the corresponding elements of the matrices on either side of the factorization $A = LL^T$. The algorithm below computes the elements of L in a column-wise fashion:

for $j = 1, \ldots, n$ **do**
  $L_{jj} = \left(A_{jj} - \sum_{k=1}^{j-1} (L_{jk})^2\right)^{1/2}$
  for $i = j+1, \cdots, n$ **do**
    $L_{ij} = \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} L_{jk}\right) / L_{jj}.$
  end
end

Observe that in the case of a tridiagonal matrix, the algorithm simplifies considerbly. We need only compute the diagonal and sub-diagonal elements:

for $j = 1, \ldots, n$ **do**
  $L_{jj} = \left(A_{jj} - \sum_{k=1}^{j-1} (L_{jk})^2\right)^{1/2}$
  $L_{j+1,j} = A_{j+1,j}/L_{jj}.$
end

Example 2.1.1. *For the matrix A in (2.11), is easily seen that* $A = LL^T$ *where*

$$L = \begin{pmatrix} \sqrt{2} & 0 & 0 & \cdots & \cdots & 0 \\ -1/\sqrt{2} & \sqrt{3/2} & 0 & \cdots & & 0 \\ 0 & -\sqrt{2/3} & \sqrt{2} & \cdots & & 0 \\ \vdots & 0 & -1/\sqrt{2} & \sqrt{3/2} & & 0 \\ \vdots & & & \ddots & \ddots & \\ 0 & & & & \alpha_{n-1,n} & \alpha_{nn} \end{pmatrix}. \tag{2.13}$$

The elements $\alpha_{n-1,n}$ and α_{nn} depend on the dimension n, i.e.,

$$\begin{cases} \alpha_{n-1,n} = -1/\sqrt{2} \ \ \textit{and} \ \ \alpha_{nn} = \sqrt{3/2}, \ \ \textit{when} \ \ n = 2k, \ \ k \in \mathbb{N}, \\ \alpha_{n-1,n} = -\sqrt{2/3} \ \ \textit{and} \ \ \alpha_{nn} = \sqrt{2} \ \ \textit{when} \ \ n = 2k+1, \ \ k \in \mathbb{N}. \end{cases}$$

Having computed the Cholesky factorizaton of the matrix A the linear system $AU = f$ reduces to

$$AU = f \Rightarrow LL^T U = f.$$

In general, consider the solution of the linear system

$$\tilde{L}Ux = b,$$

where $\tilde{L}$ and U are given by

$$\tilde{L} = \begin{pmatrix} \delta_1 & & & & \\ \gamma_2 & \delta_2 & & \mathbf{0} & \\ & \ddots & \ddots & & \\ & & \gamma_{n-1} & \delta_{n-1} & \\ \mathbf{0} & & & \gamma_n & \delta_n \end{pmatrix}, \quad U = \begin{pmatrix} \zeta_1 & \varepsilon_1 & & & \\ & \zeta_2 & \varepsilon_2 & \mathbf{0} & \\ & & \ddots & \ddots & \\ & \mathbf{0} & & \zeta_{n-1} & \varepsilon_{n-1} \\ & & & & \zeta_n \end{pmatrix}$$

Letting $y = Ux$, it suffices to solve the system

$$\tilde{L}y = f.$$

$\tilde{L}$ is a lower triangular matrix with a single nonzero sub–diagonal, so we may apply the *back-substitution* algorithm shown below:

$y_1 = f_1/\delta_1$
for $k = 2, \cdots, n$ **do**
  $y_k = (f_k - \gamma_k y_{k-1})/\delta_k$
end

Having y, we can compute the solution x by solving the upper triangular system $Ux = y$. Once again, we can apply the back-substitution algorithm, which now takes the form:

$x_n = y_n/\zeta_n$
for $k = n-1, \ldots, 1$ **do**
  $x_k = (y_x - \varepsilon_k x_{k+1})/\zeta_k$
end

We are now ready to solve the initial system $LL^T U = f$. We write $U = (U_1, \ldots, U_n)^T$ and let $y = L^T U$, so that $Ly = f$. Given the form of L in (2.13), the following algorithm computes the solution y:

$y_1 = f_1/\sqrt{2}$
for $k = 2, \ldots, n$ *with step 2* **do**
  $y_k = (f_k + (1/\sqrt{2})y_{k-1})/\sqrt{3/2}$
end
for $k = 3, \ldots, n$ *with step 2* **do**
  $y_k = (f_k + (\sqrt{2/3})y_{k-1})/\sqrt{2}$
end

Having the solution y of the system $Ly = f$, we proceed with the solution of $L^T U = y$, where L^T is an upper triangular matrix. We apply the backward substitution algorithm, which, in the case of even n is

$U_n = y_n/\sqrt{3/2}$
for $k = n-1,\dots,1$ *with step 2* **do**
  $U_k = (y_k + (1/\sqrt{2})U_{k+1})/\sqrt{2}$
end
for $k = n-2,\dots,1$ *with step 2* **do**
  $U_k = (y_k + (\sqrt{2/3})U_{k+1})/\sqrt{3/2}$
end

and when n is odd
$U_n = y_n/\sqrt{2}$
for $k = n-1,\dots,1$ *with step 2* **do**
  $U_k = (y_k + (\sqrt{2/3})U_{k+1})/\sqrt{3/2}$
end
for $k = n-2,\dots,1$ *with step 2* **do**
  $U_k = (y_k + (1/\sqrt{2})U_{k+1})/\sqrt{2}$
end

2.1.1 An upwind scheme

Consider now the one-dimensional version of a convection-diffusion equation with Dirichlet boundary conditions

$$\begin{cases} -\alpha u'' + bu' = 0, & 0 < x < L = 1, \\ u(0) = 0,\, u(1) = 1, \end{cases} \tag{2.14}$$

whose exact solution is

$$u(x) = \frac{1-e^{Rx}}{1-e^{R}}. \tag{2.15}$$

Here, R is the *Péclet number*, defined by $R = bL/a$. Given, $n \geq 1$, integer, consider a partition of $[0,L]$ with nodes $x_i = ih$, $i = 0,\dots,n+1$, where $h = L/(n+1)$.
We set $u_0 = u(0) = 0$, $u_{n+1} = u(1) = 1$, and compute approximations $u_i \approx u(x_i)$, $i = 1,\dots,n$, by replacing the first and the second order derivatives in the equation, by the approximations (2.5) and (2.4), respectively, so that

$$b\frac{u_{i+1}-u_{i-1}}{2h} - \alpha\frac{u_{i+1}-2u_i+u_{i-1}}{h^2} = 0, \qquad i = 1,\dots,n. \tag{2.16}$$

Setting $c = Rh/2$ we may write the system of equations above as

$$-(1-c)u_{i+1} + 2u_i - (1+c)u_{i-1} = 0, \qquad i = 1,\dots,n. \tag{2.17}$$

Equations (2.17) are equivalent to a tridiagonal system of equations $Au = f$, where, for the case $n = 5$, the tridiagonal matrix A is giving by

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1+c & & & \\ -1-c & 2 & -1+c & & \\ & -1-c & 2 & -1+c & \\ & & -1-c & 2 & -1+c \\ & & & -1-c & 2 \end{pmatrix}. \tag{2.18}$$

It easy to see that if backward approximations had been used to obtain approximations of the derivates, the tridiagonal matrix A, again for the case $n = 5$ would be

$$A = \frac{1}{h^2}\begin{pmatrix} 2+c & -1 & & & \\ -1-c & 2+c & -1 & & \\ & -1-c & 2+c & -1 & \\ & & -1-c & 2+c & -1 \\ & & & -1-c & 2+c \end{pmatrix}, \tag{2.19}$$

where now $c = Rh$. The choice of the discretization used depends on the sign of the coefficient b. It can be shown that the forward difference formula for approximating the derivative is appropriate when $b < 0$, while the backward difference formula is appropriate when $b > 0$. We can succinctly write both schemes as

$$b\frac{\delta_i^* u_i}{h} - \alpha\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = 0, \qquad i = 1, \ldots, n, \tag{2.20}$$

by adopting the notational convention

$$\delta_i^* = \begin{cases} \delta_i^- & \text{if } b > 0, \\ \delta_i^+ & \text{if } b < 0. \end{cases} \tag{2.21}$$

Using the Upwind Scheme code we can verify that this discetization produces accurate approximations of the exact solution (2.15) of the convection–diffusion equation (2.14).

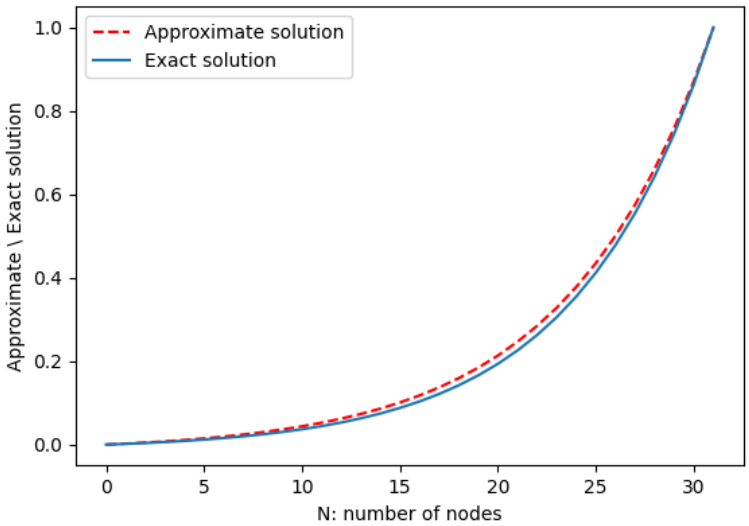

Figure 2.1: Finite difference solution and exact solution of the convection-diffusion equation (2.14) with $a = 2$, $b = 9$ and $N = 30$

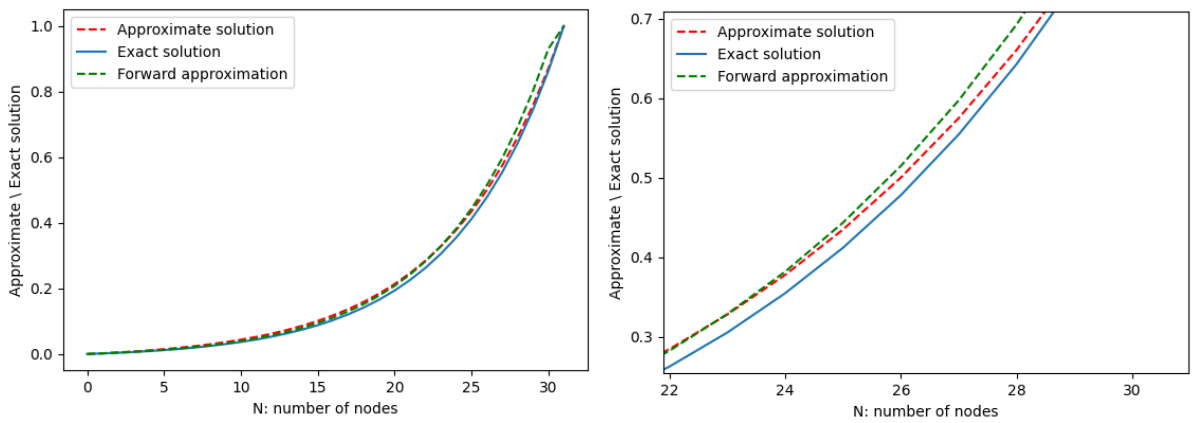

Figure 2.2: A comparison between the upwind scheme choice and the forward approximation, in the case of $b > 0$

The figure above presents the convergence of the approximation solution to the exact solution using the

upwind scheme (red line) and the forward approximation (green line) in the case of $b > 0$. Clearly, the upwind scheme provides a more accurate approximation.

2.1.2 Cyclic Reduction

Consider now the two-dimensional problem

$$-\left(\frac{\partial^2 u}{\partial x_1^2}+\frac{\partial^2 u}{\partial x_2^2}\right)=f \qquad \text{in } \Omega, \tag{2.22}$$

with the boundary condition

$$u=0 \quad \text{on } \Gamma=\partial\Omega, \tag{2.23}$$

where $\Omega=(0,l_1)\times(0,l_2)$. We consider a uniform partition of $\Omega=(0,l_1)\times(0,l_2)$ with $(n_1+1)\times(n_2+1)$ points.

$$x_{1,i}=i\times h_1, \quad i=0,\dots,n_1+1, \qquad x_{2,j}=j\times h_2, \quad j=0,\dots,n_2+1,$$

where

$$h_1=\frac{l_1}{n_1+1}, \qquad h_2=\frac{l_2}{n_2+1}.$$

If we use the second order finite difference scheme (2.4) for the discretization of the partial derivatives, in (2.22) we obtain a symmetric, block–diagonal system of equations for the approximations $u_{ij}\approx u(x_{1,i},x_{2,j})$. The system matrix A, when $n_1=3$ and $n_2=5$, is

$$A=\frac{1}{h^2}\begin{pmatrix} B & -I & \\ -I & B & -I \\ & -I & B \end{pmatrix}, \qquad B=\begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & -1 & 4 & -1 \\ & & -1 & 4 \end{pmatrix}.$$

We describe the method of *cyclic reduction* for solving a block-tridiagonal system of equations of the form

$$\frac{1}{h^2}\begin{pmatrix} B & -I & \\ -I & B & -I \\ & -I & B \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}=\begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \tag{2.24}$$

where u_i, f_j for $i=1,2,3$ and $j=1,2,3$, are vectors of length 4.

To this end, we multiply the second row by B and add to it the first and the third rows, so that

$$\frac{1}{h^2}\left(B^2-2I\right)u_2=(Bf_2+f_1+f_3), \tag{2.25}$$

which we can easily solve. Having u_2 we obtain u_1 and u_3 by solving similar tridiagonal systems. This procedure may be generalized to $n\times n$ block-tridiagonal systems with $n=2^p-1$ using the so-callled Buneman variant, see [2] Let us assume that the matrix A has the form

$$\begin{pmatrix} D & F & & & \\ F & D & F & & \\ & \ddots & \ddots & \ddots & \\ & & F & D & F \\ & & & F & D \end{pmatrix}, \tag{2.26}$$

where F and D are $q\times q$ matrices that satisfy $DF=FD$ and where $n=2^k-1$. Notice that these conditions hold for the discretization of Poisson's equations described above. The integer n is determined by the size of the mesh and can often be chosen to be of the particular form.
The basic ideas behind cyclic reduction is to halve the dimension of the problem repeatedly until we are

left with a single $q \times q$ system for the q-vector $x_{2^{k-1}}$. The system is then solved by standard means. The previously eliminated x_i are found by a back-substitution process. For example, in the case $n = 7$ we have

$$
\begin{aligned}
Dx_1 + Fx_2 &= b_1, && (2.27)\\
Fx_1 + Dx_2 + Fx_3 &= b_2, && (2.28)\\
Fx_2 + Dx_3 + Fx_4 &= b_3, && (2.29)\\
Fx_3 + Dx_4 + Fx_5 &= b_4, && (2.30)\\
Fx_4 + Dx_5 + Fx_6 &= b_5, && (2.31)\\
Fx_5 + Dx_6 + Fx_7 &= b_6, && (2.32)\\
Fx_6 + Dx_7 &= b_7, && (2.33)
\end{aligned}
$$

For $i = 2,4,6$, we multiply equations $i-1, i$ and $i+1$ by $F, -D$, and F, respectively and add the resulting equation to obtain

$$
\begin{aligned}
(2F^2 - D^2)x_2 + F^2 x_4 &= F(b_1 + b_3) - Db_2,\\
F^2 x_2 + (2F^2 - D^2)x_4 + F^2 x_6 &= F(b_3 + b_5) - Db_4,\\
F^2 x_4 + (2F^2 - D^2)x_6 &= F(b_5 + b_7) - Db_6.
\end{aligned}
$$

This is a reduced block tridiagonal system of the form

$$
\begin{aligned}
D^{(1)}x_2 + F^{(1)}x_4 &= b_2^{(1)},\\
F^{(1)}x_2 + D^{(1)}x_4 + F^{(1)}x_6 &= b_2^{(1)},\\
F^{(1)}x_4 + D^{(1)}x_6 &= b_6^{(1)},
\end{aligned}
$$

where $D^{(1)} = 2F^2 - D^2$ and $F^{(1)} = F^2$ commute. Applying the same elimination strategy as above, we multiply these three equations respectively by $F^{(1)}$, $-D^{(1)}$, and $F^{(1)}$. When these equations are added together , we obtain a single equation

$$
\left(2\left[F^{(1)}\right]^2 - \left[D^{(1)}\right]^2\right)x_4 = F^{(1)}\left(b_2^{(1)} - b_6^{(1)}\right) - D^{(1)}b_4^{(1)},
$$

which we write as

$$
D^{(2)}x_4 = b_4^{(2)}.
$$

This completes the cyclic reduction. We now solve this small $q \times q$ system for x_4. The vectors x_2 and x_6 are found by solving the systems

$$
\begin{aligned}
D^{(1)}x_2 &= b_2^{(1)} - F^{(1)}x_4,\\
D^{(1)}x_6 &= b_6^{(1)} - F^{(1)}x_4.
\end{aligned}
$$

Finally, we use the first, third, fifth and seventh equations in (2.27) – (2.33) to compute x_1, x_3, x_5, and x_7, respectively.

2.2 The coordinate format

The storage of large sparse matrices is especially important for the performance of direct (and iterative) solvers. Several storage schemes have been devised for storing sparse matrices.
The simplest storage scheme for sparse matrices is the so-called “coordinate format”. This scheme uses three arrays:

- An array AA containing all the nonzero elements of the sparse matrix A, in any order.
- An integer array JR containing their row indices.
- A second integer array JC containing their column indices.

All arrays are of length N_z, where N_z is the number of nonzero elements of A.

Example 2.2.1. *The sparse matrix*

$$A = \begin{pmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{pmatrix} \tag{2.34}$$

will be represented by

AA	12	9	7	5	1	2	11	3	6	4	8	10
JR	5	3	3	2	1	1	4	2	3	2	3	4
JC	5	5	3	4	1	4	4	1	1	2	4	3

In the example above, the element order is arbitrary. However, usually the elements are stored either by row or by column. If the elements were listed by row, the array JR would not be needed and could be replace by an array containing the indices of the beginning of each row in AA. The resulting format is the so-called *Compressed Sparse Row (CSR)* format. As before, we set the three arrays:

- An array AA containing all the nonzero elements of A stored row by row.
- A second integer array JA containing their column indices of α_{ij} as stored in AA.
- An integer array IA contains the pointers to the beginning of each row in the arrays AA and JA. Thus, if AR(k) $= \alpha_{ij}$ and JA(k) $= j$, then for $i \in \{1, \ldots, n\}$, where n is the dimension of the matrix A we have
  - IA(i) = k, where α_{ij} in AR(k) is the first element stored in AR for row i.
  - IA(i) = IA(i+1) if all $\alpha_{ij} = 0$ in row i
  - IA(n+1) = $N_z + 1$.

The length of AA and JR is N_z while the length of IA is $n+1$.

Example 2.2.2. *The above matrix may be stored as follows:*

AA	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
JA	*1*	*4*	*1*	*2*	*4*	*1*	*3*	*4*	*5*	*3*	*4*	*5*
IA	*1*	*3*	*6*	*10*	*12*	*13*						

A variation of the CSR format is the *Compressed Sparse Column (CSC)* scheme, where the elements are listed column by column. Another variation is the *Modified Sparse Row (MSR)* format. There are only two matrices where,

- The matrix AA which has, at the first n positions, the diagonal elements of the matrix, in order. The position $n+1$ is empty and the rest of nonzero elements of A, distributed among the rest of positions.

- The matrix JA, whose elements distribute as follows: The $n+1$ positions contain the pointer to the beginning of each row in AA and JA and for the rest of positions contain the corresponding column of each element of AA(k) for $k > n+1$.

Example 2.2.3. *For the matrix* (2.34)*, the AA and JA according to MSR, are:*

$$AA \quad \boxed{1 \quad 4 \quad 7 \quad 11 \quad 12 \quad * \quad 2 \quad 3 \quad 5 \quad 6 \quad 8 \quad 9 \quad 10}$$

$$JA \quad \boxed{7 \quad 8 \quad 10 \quad 13 \quad 14 \quad 14 \quad 4 \quad 1 \quad 4 \quad 1 \quad 4 \quad 5 \quad 3}$$

*where * denoted the empty posistion.*

Another storage format appropriate for diagonally structured matrices, that is matrices with nonzero elements along a small number of diagonals is as follows: the diagonals are stored in a rectangular array DIAG(1:n, 1:Nd), where Nd is the number of diagonals. The number of offsets of each diagonal is stored in an second integer array IOFF(1:Nd).

Example 2.2.4.

$$A = \begin{pmatrix} 1 & 0 & 2 & 0 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 0 & 6 & 7 & 0 & 8 \\ 0 & 0 & 9 & 10 & 0 \\ 0 & 0 & 0 & 11 & 12 \end{pmatrix}$$

will be represented by the two arrays:

$$DIAG = \boxed{\begin{array}{ccc} * & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & * \\ 11 & 12 & * \end{array}} \qquad IOFF = \boxed{-1 \quad 0 \quad 2}$$

2.3 Solving sparse linear system using Python

Systems or linear equations with sparse coefficient matrices may be solved easily using Python's **SciPy** library, and more specifically the routines in the sub-packages **scipy.sparse** and **scipy.sparse.linalg**. A short review of the available utility functions and solvers using the default CSC format is presented here. The solution process for matrices stored tin the CSR format is presented in A.2. For both storage schemes, the solution of a linear system is accomplished with the command **spsolve**$(*,*)$.

2.3.1 Package Numpy

From the package **Numpy** we use the commands:

- **np.size(A, axis=0)** $\longrightarrow$ Gives the number of rows of the matrix A
- **np.ones(i)** $\longrightarrow$ Gives a vector with all elements equal to 1 and length i

2.3.2 Sparse matrices (scipy.sparse)

To build the coefficient matrix, we may use functions provided in the subpackage **scipy.sparse**. There functions to define a sparse identity matrix, or an upper tridiagonal matrix, and several others:

- **eye(m[, n, k, dtype, format])** $\longrightarrow$ Sparse matrix with ones on the diagonal
- **identity(n[, dtype, format])** $\longrightarrow$ Identity matrix in sparse format
- **diags(diagonals[, offsets, shape, format, dtype])** $\longrightarrow$ Construct a sparse matrix from diagonals

- **block_diag(mats[, format, dtype])** $\longrightarrow$ Build a block diagonal sparse matrix from the provided blocks.

- **tril(A[, k, format])** $\longrightarrow$ Return the lower triangular part of a matrix in sparse format.

- **triu(A[, k, format])** $\longrightarrow$ Return the upper triangular part of a matrix in sparse format.

Given a sparse matrix, to find the values of its nonzero elements we could use the function:

- **find(A)** $\longrightarrow$ Return the indices and values of the nonzero elements of a matrix.

We could also confirm that a matrix is a sparse matrix with the commands:

- **issparse(x)**

- **isspmatrix(x)**

2.3.3 Sparse Linear Algebra (scipy.sparse.linalg)

We can easily find the solution of a system by using the command **spsolve**. This command returns the exact solution of a system, where the coefficient matrix is assumed to be stored in the CSC format. However, converting the storage scheme to CSR, as is shown in the examples below, usually results tin faster solution times.

2.3.4 Applications and examples

We consider five symmetric, positive definite matrices with corresponding sizes 4884×4884, 10974×10974, 15439×15439, 23052×23052 and 30401×30401. The first four matrices arise from a structural problem while the last one arises from a random problem (more details about them my be found at the site Suite Sparse Matrix Collection). We can see, graphically, their sparsity structure of these matrices using the command **plt.spy**:

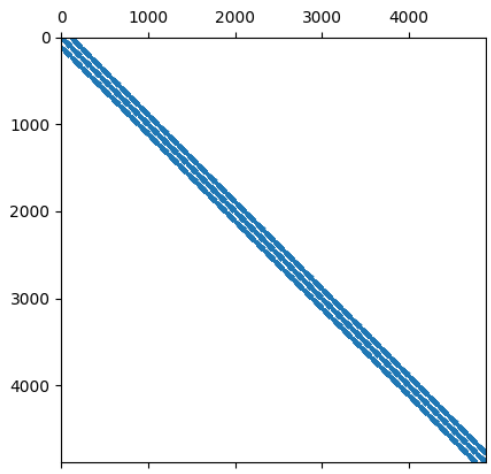

Figure 2.3: Graph of bcsstk16

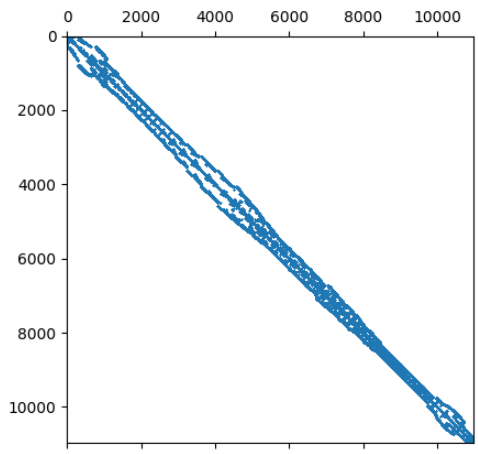

Figure 2.4: Graph of bcsstk17

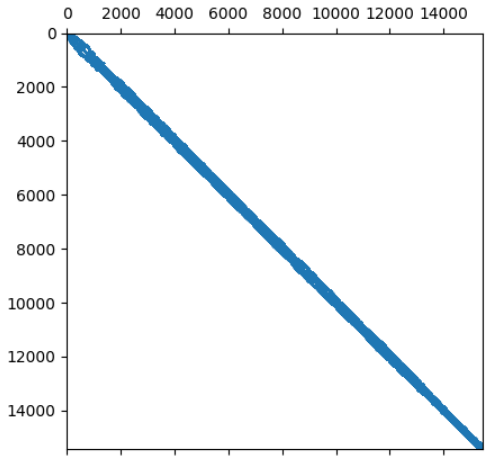

Figure 2.5: Graph of bcsstk25

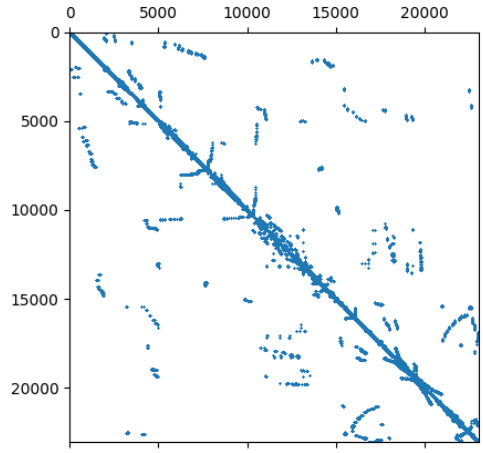

Figure 2.6: Graph of bcsstk36

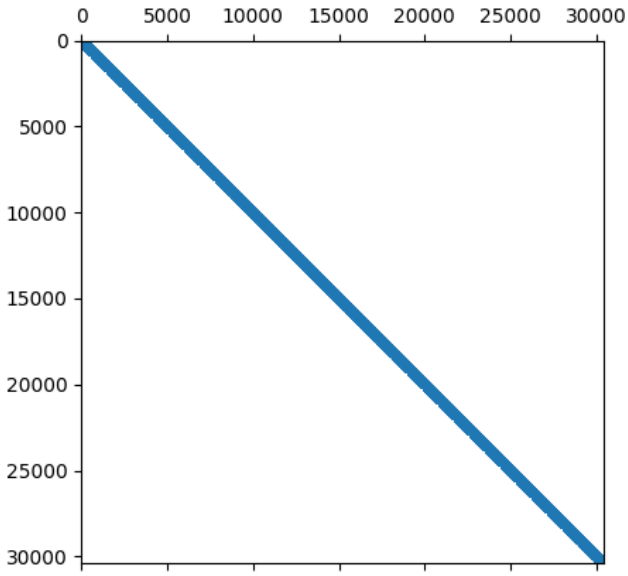

Figure 2.7: Graph of wathen100

Using the Python code A.2 we obtain the solution of the system $Ax = b$, where the right–hand side b is the unit vector. The solution x and the time needed to find the solution of the system $Ax = b$ by the code A.2 are shown below:

```
Matrix: bcsstk25.mat
Non-zero elements: 252241
Solve in  2.0612461  seconds
[ 2.03384805e-06  2.49393362e-06  9.56244567e-09 ...  5.73893861e-07
 -6.26957911e-07 -3.70575792e-07]

Next matrix:
Matrix: bcsstk16.mat
Non-zero elements: 290378
Solve in  0.31377960000000016  seconds
[ 3.89719769e-08 -1.50445471e-08  1.00000000e+00 ...  6.81791603e-07
  2.67834286e-07  1.00000000e+00]

Next matrix:
Matrix: bcsstk17.mat
Non-zero elements: 428650
Solve in  0.4342156999999993  seconds
[ 1.00000000e+00  9.28716555e-03  2.11548110e-04 ...  5.26814471e-05
 -1.64567602e-06  3.49673068e-04]

Next matrix:
Matrix: wathen100.mat
Non-zero elements: 471601
Solve in  0.7952624000000004  seconds
[0.14676698 0.04238437 0.13706134 ... 0.10417945 0.09707795 0.25494121]

Matrix: bcsstk36.mat
Non-zero elements: 1143140
Solve in  9.1718284  seconds
[-2.53417062e-04 -9.27900398e-05  1.51942724e-04 ... -3.49279059e-04
 -5.26947249e-02 -1.55982185e-02]
```

Figure 2.8: Results of run code A.2

We normalize the time using as unit time the required time for the matrix with the least number non zero elements, that is the matrix labeled 'bcsstk25'. The graph below shows the relation between time (in seconds) and the number of non-zero elements.

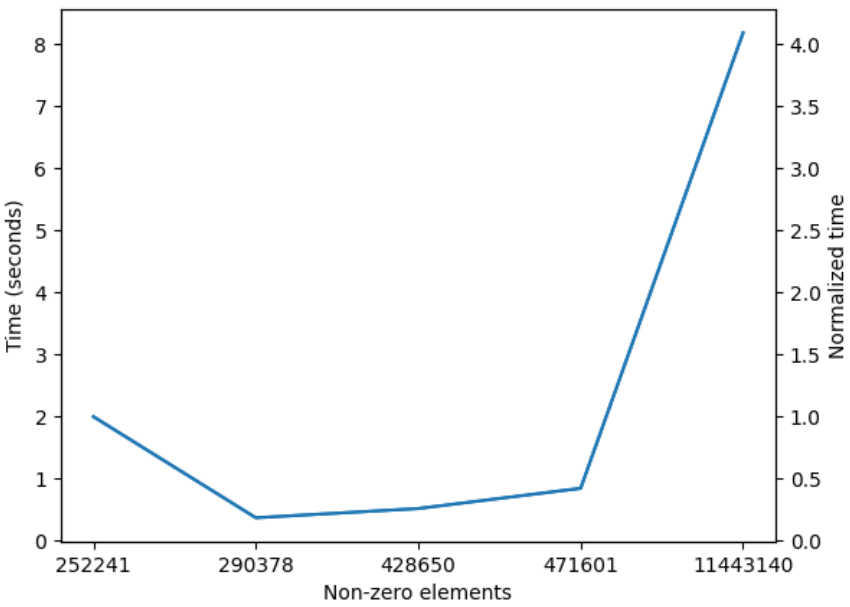

Figure 2.9: Comparison plot between time (also normalized time) and non-zero elements

CHAPTER 3

Stationary Iterative methods

There are several differences between direct and iterative methods for the solution of systems of linear equations. Two of the main differences are detailed below: First, to obtain the solution of a system of linear equations using a direct method means that we should perform a fixed amount of floating point operations, depending only on the size of system matrix. For example, the LU factorization method requires $O(n^3)$ floating point operations, with n the number of equations. Moreover, in the case of a sparse matrix the elementary row operations used in the LU factorization may introduce nonzero elements in the system matrix. By contrast, if we use an iterative method, like Jacobi, then the matrix is not changed and we can stop iterating when a sufficiently accurate solution has been obtained. Secondly, for a direct method we need to store the *entire* matrix while with an iterative method, only matrix–vector products are usually required

3.1 Stationary Iterative Methods

Suppose that we have the system $Ax = b$, where $A \in \mathbb{R}^{n,n}$, $b \in \mathbb{R}^n$ and $\det(A) \neq 0$. *Stationary Iterative Methods* to solve this system, start from an initial guess of the solution and compute approximations $x^{(k)}$ for $k = 1, 2 \ldots$, using particular matrix splittings. The two best known Stationary Iterative Methods are the *Jacobi* and *Gauss–Seidel* methods. Both of them require that the matrix has nonzero diagonal entries. Observe that in this case, the system $Ax = b$ may be written as

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j - \sum_{j=i+1}^{n} a_{ij} x_j \right), \quad i = 1, \ldots, n. \tag{3.1}$$

3.1.1 A short description of Jacobi and Gauss-Seidel Methods

Jacobi's Method is based on the following strategy: starting from a given initial guess of the solution, $x^{(0)} \in \mathbb{R}^n$, and substituting it into the right side of the equation (3.1) we can find another approximation $x^{(1)}$. Similarly, using the last approximation we can find a, hopefully, better approximation to the solution. This procedure stops when the error $e_m = \|x^{(m)} - x\|$ is small enough. Therefore, the equation (3.1) for the $(m+1)$-st approximation is

$$x_i^{(m+1)} := \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(m)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(m)} \right), \qquad i = 1, \ldots, n. \tag{3.2}$$

Formula (3.2) shows that $x_i^{(m+1)}$ depends only on the previous iterates $x_j^{(m)}$, $j = 1,\dots,i-1,i+1,\dots,n$, hence the computation for every j can be computed in parallel, using n different compute units with the same data, $A, b, x^{(m)}$.

Another approach for the estimation of $x_i^{(m+1)}$ is the *Gauss-Seidel Method* for which

$$x_i^{(m+1)} := \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(m)}\right), \quad i = 1,\dots,n. \tag{3.3}$$

Observe that the value of $x_i^{(m+1)}$, except for x_j^m, depends on $x_j^{(m+1)}$ which have already been computed. At first glance, the Gauss-Seidel Method seems to approach faster the solution since $x_j^{(m+1)}$ comprises a better approximation than $x_j^{(m)}$, $j = 1,\dots,i-1$. However, there are systems for which Jacobi's method converges while Gauss-Seidel does not.

3.1.2 Matrix splittings for the Jacobi and the Gauss–Seidel methods

Suppose that

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

and define the matrices D, L, U by

$$D = \begin{pmatrix} a_{11} & & & & \\ & a_{22} & & \mathbf{0} & \\ & & a_{33} & & \\ & \mathbf{0} & & \ddots & \\ & & & & a_{nn} \end{pmatrix}, \quad L = \begin{pmatrix} 0 & & & & \\ a_{21} & 0 & & \mathbf{0} & \\ a_{31} & a_{32} & 0 & & \\ \vdots & \vdots & & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0 \end{pmatrix},$$

$$U = \begin{pmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ & 0 & a_{23} & \cdots & a_{2n} \\ & & 0 & & \vdots \\ & \mathbf{0} & & \ddots & a_{n-1,n} \\ & & & & 0 \end{pmatrix}$$

so the $A = D + L + U$. According to above definition of D, L and U, equations (3.2) and (3.3) may be written as

$$\begin{aligned} Dx^{(m+1)} &= -(L+U)x^{(m)} + b, \\ (L+D)x^{(m+1)} &= -Ux^{(m)} + b, \end{aligned}$$

respectively, or, equivalently

$$x^{(m+1)} = -D^{-1}(L+U)x^{(m)} + D^{-1}b, \tag{3.4}$$

$$x^{(m+1)} = -(L+U)^{-1}Ux^{(m)} + (L+U)^{-1}b. \tag{3.5}$$

3.1.3 Convergence of iterative methods

We assume that $x^{(m+1)}$ is given by

$$Mx^{(m+1)} = Nx^{(m)} + b, \qquad A = M - N \tag{3.6}$$

$$x^{(m+1)} = M^{-1}Nx^{(m)} + M^{-1}b, \qquad m \in \mathbb{N}_0 \tag{3.7}$$

where M is an invertible matrix and N is such that $A = M - N$.

Remark: Observe that $M_J = D$, $N_J = -(L+U)$ for Jacobi's method, and $M_{GS} = L+D$, $N_{GS} = -U$ for Gauss-Seidel method.

Suppose now that the current approximation $x^{(k)}$ is the exact solution x, then $x^{(k+1)}$ should be also be the exact solution. Therefore, by (3.7) it should be true that

$$\begin{aligned} & x = M^{-1}Nx + M^{-1}b \\ \Rightarrow \quad & Mx = Nx + b \\ \Rightarrow \quad & (M-N)x = b \end{aligned} \tag{3.8}$$

Since $A = M - N$, it is obvious that if the sequence $x^{(m)}$ converges, its limit is the solution of the system $Ax = b$. By subtracting (3.8) from (3.6), we get

$$x^{(m+1)} - x = G(x^{(m)} - x), \quad m \in \mathbb{N}_0, \tag{3.9}$$

where G is the so-called *iteration matrix* and is defined as

$$G = M^{-1}N.$$

It is easy to see that if $x^{(m)} - x$ is the current error $e^{(m)}$ then from (3.9) we get

$$\begin{aligned} e^{(m)} &= Ge^{(m-1)} = G(Ge^{(m-2)}) = \cdots = G^m e^{(0)} \\ x^{(m)} &- x = G^m(x^{(0)} - x) \end{aligned} \tag{3.10}$$

Suppose that $\|\cdot\|$ is an induced matrix norm. From the last equation we have

$$\|e^{(m)}\| = \|x^{(m)} - x\| \leq \|G^m\| \, \|x^{(0)} - x\| = \|G^m\| \, \|e^{(0)}\|, \quad m \in \mathbb{N}_0. \tag{3.11}$$

There are initial vectors $x^{(0)}$, for all values of m, that satisfy the above with equality. Thus, we come to the conclusion that the sequence $(x^{(m)})_{m\in\mathbb{N}_0}$ converges to x if and only if

$$\lim_{m\to\infty} G^m = 0 \implies \lim_{m\to\infty} \|G^m\| = 0, \tag{3.12}$$

for every induced matrix norm. With the purpose of studying convergence, we will recall the following definition of the spectral radius:

Definition 3.1. *The spectral radius of a matrix $P \in \mathbb{C}^{n,n}$ is defined as*

$$\rho(P) := max_{1\leq i\leq n} |\lambda_i(P)|, \tag{3.13}$$

where $\lambda_i = \lambda_i(P)$ are the eigenvalues of P.

We have the following:

Lemma 3.1. *Assume that $\|\cdot\|$ is a norm in $\mathbb{C}^n$. Then, for every $P \in \mathbb{C}^{n,n}$, it is true that*

$$\rho(P) \leq \|P\|. \tag{3.14}$$

Conversely, for every $P \in \mathbb{C}^{n,n}$ and $\varepsilon > 0$, there is a norm $\|\cdot\|$ in $\mathbb{C}^n$ such that

$$\|P\| \leq \rho(P) + \varepsilon. \tag{3.15}$$

Proof. We refer the reader to [11], section 3.9, for a proof. □

Definition 3.2. *We say that* $\{B_k\}_{k=1}^{\infty}$ ***converges*** *if and only if there is a matrix* $B \in \mathbb{C}^{m\times n}$ *such that, for every* $\varepsilon > 0$ *there exists an integer* $K > 0$ *such that, if* k *is any integer satisfying* $k \geq K$*, then it follows that*

$$\|B_k - B\|_{max} < \varepsilon,$$

where the norm $\|\cdot\|$ *may be any norm of* $\mathbb{C}^{m\times n}$ *(because of the equivalence of norms). We write* $B_k \longrightarrow B$*. We say that the square matrix* $A \in \mathbb{C}^{n\times n}$ *is* ***convergent to zero*** *if and only if* $A^k \longrightarrow \mathbf{0} \in \mathbb{C}^{n\times n}$*.*

Considering last definition and using Lemma 3.1, it ia easy to prove the next result

Theorem 3.1. *Let* x *be the solution of the system* $Ax = b$*. The following are equivalent:*

(a) Iterative method (3.7) converges and this means that for every $x^{(0)} \in \mathbb{C}^n \Rightarrow x^{(m)} \xrightarrow{m\to\infty} x$*.*

(b) $\rho(G) < 1$*, where* G *the ireration matrix* $G = M^{-1}N$ *of (3.7).*

(c) There is an induced matrix norm $|||\cdot|||$ *such that* $|||G||| < 1$*.*

(d) $\lim_{m\to\infty} G^m = 0$*.*

Proof. $\underline{(\mathbf{a}) \Longrightarrow (\mathbf{b})}$: From (a) we know that $x^{(m)} \xrightarrow{m\longrightarrow\infty} x$. According to 3.10

$$G^m(x^{(0)} - x) \xrightarrow{m\longrightarrow\infty} 0, \qquad x \in \mathbb{C}^n. \tag{3.16}$$

For any given $y \in \mathbb{C}^n$, let $x^{(0)} = y + x$. Thus, the (3.16) can be written as

$$G^m y \xrightarrow{m\longrightarrow\infty} 0, \qquad \forall y \in \mathbb{C}^n. \tag{3.17}$$

Suppose that λ is an eigenvalue of G and z its eigenvector. Then, $G^m z = \lambda^m z, \quad m \in \mathbb{N}_0$. However, $G^m z \xrightarrow{m\to\infty} 0$, then for any norm $\|.\|$ in $\mathbb{C}^n$:

$$\|G^m z\| \to 0 \Rightarrow |\lambda|^m \|z\| \to 0 \Rightarrow |\lambda| < 1 \tag{3.18}$$

So, $\rho(G) = \max_i |\lambda_i(G)| < 1$.

$\underline{(\mathbf{b}) \Longrightarrow (\mathbf{c})}$: Suppose $\rho(G) = \max|\lambda_i(G)| < 1$ and ε a positive number such that $0 < \varepsilon < 1 - \rho(G)$. According to lemma 3.1, there is an induced matrix norm $|||\cdot|||$ in $\mathbb{C}^{n,n}$ such that $|||G||| \leq \rho(G) + \varepsilon < 1$.

$\underline{(\mathbf{c}) \Longrightarrow (\mathbf{d})}$: If $|||G||| < 1$, on the grounds that $|||G^m||| = |||G \cdot G \cdots G||| \leq |||G|||^m$, we have

$$|||G^m||| \longrightarrow 0 \Longleftrightarrow G^m \xrightarrow{m\longrightarrow\infty} 0. \tag{3.19}$$

$\underline{(\mathbf{d}) \Longrightarrow (\mathbf{a})}$: If $G^m \xrightarrow{m\longrightarrow\infty} 0$ for any norm $\|.\|$ in $\mathbb{C}^n$ then, because of (3.10), we have

$$x^{(m)} \xrightarrow{m\longrightarrow\infty} 0 \quad \forall x^{(0)} \in \mathbb{C}^n.$$

□

Most of the time, initial approximation $x^{(0)}$ is chosen arbitrarily. In order to terminate the iterative method, it's important to define a tolerance $\varepsilon > 0$ such that

$$\|x^{(N)} - x^{(N-1)}\| \leq \varepsilon \tag{3.20}$$

As a consequence, we can prove that if the inequality (3.20) holds, then it's true that

$$\|x - x^{(N)}\| \leq \frac{\varepsilon\sigma}{1-\sigma},$$

with $\sigma = \|G\|$.

Proposition 3.1. *For any $\varepsilon > 0$, the inequality $\|x - x^{(N)}\| \leq \frac{\varepsilon\sigma}{1-\sigma}$ follows from $\|x^{(N)} - x^{(N-1)}\| \leq \varepsilon$, where $\|G\| = \sigma < 1$.*

Proof. Let the function $\phi(x^{(N)}) = x^{(N+1)} = Gx^{(N)} + M^{-1}b$, where $x^{(N)}$ is the $N-th$ iteration of the iterative method. We have the following inequality:

$$\begin{aligned}
\|\phi(x^{(N)}) - \phi(x^{(N-1)})\| &= \|Gx^{(N)} + M^{-1}b - Gx^{(N-1)} - M^{-1}b\| \\
&= \|G(x^{(N)} - x^{(N-1)})\| \\
&\leq \|G\|\|x^{(N)} - x^{(N-1)}\|
\end{aligned}$$

We have also the relations:

$$\begin{aligned}
\|x^{(2)} - x^{(1)}\| &= \|\phi(x^{(1)}) - \phi(x^{(0)})\| \leq \|G\|\|x^{(1)} - x^{(0)}\| \\
\|x^{(3)} - x^{(2)}\| &= \|\phi(x^{(2)}) - \phi(x^{(1)})\| \leq \|G\|\|x^{(2)} - x^{(1)}\| \leq \|G\|^2\|x^{(1)} - x^{(0)}\| \\
&\vdots \\
\|x^{(N+1)} - x^{(N)}\| &= \|\phi(x^{(N)}) - \phi(x^{(N-1)})\| \leq \cdots \leq \|G\|^N\|x^{(1)} - x^{(0)}\|
\end{aligned}$$

So for $k \in \mathbb{N}$

$$\begin{aligned}
\|x^{(N+k)} - x^{(N)}\| &\leq \|x^{(N+k)} - x^{(N+k-1)}\| + \|x^{(N+k-1)} - x^{(N)}\| \\
&\leq \|x^{(N+k)} - x^{(N+k-1)}\| + \|x^{(N+k-1)} - x^{(N+k-2)}\| + \|x^{(N+k-2)} - x^{(N)}\| \\
&\leq \cdots \\
&\leq \|x^{(N+k)} - x^{(N+k-1)}\| + \|x^{(N+k-1)} - x^{(N+k-2)}\| + \cdots + \|x^{(N+1)} - x^{(N)}\| \\
&\leq \|G\|^{N+k-1}\|x^{(1)} - x^{(0)}\| + \|G\|^{N+k-2}\|x^{(1)} - x^{(0)}\| + \cdots + \|G\|^N\|x^{(1)} - x^{(0)}\| \\
&= \sigma^N(1 + \sigma + \sigma^2 + \cdots + \sigma^{k-1})\|x^{(1)} - x^{(0)}\| \\
&= \sigma^N\frac{1-\sigma^k}{1-\sigma}\|x^{(1)} - x^{(0)}\|
\end{aligned}$$

Therefore, the following inequality holds

$$\|x^{(N+k)} - x^{(N)}\| \leq \frac{\sigma^N}{1-\sigma}\|x^{(1)} - x^{(0)}\|. \tag{3.21}$$

Now define the continuous function $g(x) := \|x - x^{(N)}\|$. Because of continuity we have

$$\|x - x^{(N)}\| = g(x) = g(\lim_{k\to\infty} x^{N+k}) = \lim_{k\to\infty} g(x^{(N+k)}) = \lim_{k\to\infty} \|x^{(N+k)} - x^N\| \leq \frac{\sigma^N}{1-\sigma}\|x^{(1)} - x^{(0)}\|.$$

Define

$$y^{(0)} = x^{(N-1)} \quad \text{and} \quad y^{(1)} = \phi(y^{(0)}) = \phi(x^{(N-1)}) = x^N.$$

From (3.21) we have

$$\|y^{(1)} - x\| \leq \frac{\sigma}{1-\sigma}\|y^{(1)} - y^{(0)}\| = \frac{\sigma}{1-\sigma}\|x^{(N)} - x^{(N-1)}\| \leq \frac{\varepsilon\sigma}{1-\sigma}.$$

Therefore,

$$\|x^{(N)} - x\| \leq \frac{\varepsilon\sigma}{1-\sigma}$$

which completes the proof. □

Considering the convergence criteria for the general stationary iterative method above, we must find necessary conditions for the matrix A in order that either (b) or (c) of (3.1) are true. A particularly useful is the so called *diagonal dominance*. We recall that a square matrix A is called *strictly diagonally dominant* if

$$|\alpha_{ii}| > \sum_{\substack{j=1\\ j\neq i}}^{n} |\alpha_{ij}|, \qquad 1 \le i \le n, \tag{3.22}$$

where we assume that $\alpha_{ii} \neq 0$, for all $1 \le i \le n$.

Proposition 3.2. *Let A be a strictly diagonally dominant matrix. Then*

(i) A is invertible with nonzero elements in the main diagonal.

(ii) Iteration matrices $G_J = -D^{-1}(L+U)$, $G_{GS} = -(L+U)^{-1}U$ of the methods Jacobi and Gauss-Seidel, respectively, satisfy the inequalities $\|G_J\|_\infty < 1$, $\|G_{GS}\|_\infty < 1$.

(iii) The Jacobi and Gauss-Seidel methods converge.

Proof. (**i**) : Let λ an eigenvalue of A and $w \in \mathbb{C}^n$ the corresponding eigenvector. Then, from $Aw - \lambda w$, we get:

$$\sum_{j=1}^{n} \alpha_{ij} w_j = \lambda w_i, \qquad 1 \le i \le n,$$

that is

$$(\alpha_{ii} - \lambda) w_i = -\sum_{\substack{j=1\\ j\neq i}}^{n} \alpha_{ij} w_j, \qquad 1 \le i \le n.$$

As a result,

$$|\alpha_{ii} - \lambda||w_i| \le \sum_{\substack{j=1\\ j\neq i}}^{n} |\alpha_{ij}||w_j|, \qquad 1 \le i \le n. \tag{3.23}$$

Let s, $1 \le s \le n$, an index for which,

$$|w_s| = \max_{1\le k\le n} |w_k|.$$

Obviously, $w_s \neq 0$. From (3.23) with $i = s$, it follows that

$$|\alpha_{ss} - \lambda| \le \sum_{\substack{j=1\\ j\neq s}}^{n} |\alpha_{sj}| \frac{|w_j|}{|w_s|} \le \sum_{\substack{j=1\\ j\neq s}}^{n} |\alpha_{sj}|.$$

To sum up, we have shown that for every eigenvalue λ there is s, $1 \le s \le n$, such that

$$|\alpha_{ss} - \lambda| \le \sum_{\substack{j=1\\ j\neq s}}^{n} |\alpha_{sj}| \tag{3.24}$$

This inequality is called *Gerschgorin's inequality*. Thus, if there is a zero eigenvalue, then the inequality (3.24) will provide a contradiction, because of (3.22). In conclusion the matrix A has only nonzero eigenvalues, so it is invertible.

(**ii**) : Define C as

$$C := \max_{1\le i\le n} \frac{1}{|\alpha_{ii}|} \tag{3.25}$$

According to (3.22), $C < 1$. We will show that $\|G_J\|_\infty \le C$ and $\|G_{GS}\|_\infty \le C$. In the case of Jacobi's method, we have

$$\|G_J\|_\infty = \|D^{-1}(L+U)\|_\infty = \max_{1\le i\le n} \sum_{\substack{j=1\\ j\neq i}}^{n} \frac{|\alpha_{ij}|}{|\alpha_{ii}|} < \max_{1\le i\le n} \frac{|\alpha_{ii}|}{|\alpha_{ii}|} = 1. \tag{3.26}$$

In the case of Gauss-Seidel method, we consider the vector $u = G_{GS}y$, for $y \in \mathbb{R}^n$, which satisfies

$$(L+U)u = -Uy$$

so that,

$$u_i = \frac{1}{\alpha_{ii}}\Big\{-\sum_{j=1}^{i-1}\alpha_{ij}u_j - \sum_{j=i+1}^{n}\alpha_{ij}y_j\Big\}, \qquad i = 1,\dots,n. \tag{3.27}$$

We will prove by induction that

$$|u_i| \le C\|y\|_\infty, \qquad 1 \le i \le n. \tag{3.28}$$

For $i = 1$

$$|u_1| = \frac{1}{|\alpha_{11}|}\Big|\sum_{j=2}^{n}\alpha_{1j}y_j\Big| \le C\|y\|_\infty. \tag{3.29}$$

Suppose now that the claim is true for $1,\dots,i-1$. Then

$$|u_i| \le \frac{1}{|\alpha_{ii}|}\Big\{\sum_{j=1}^{i-1}|\alpha_{ij}||u_j| + \sum_{j=i+1}^{n}|\alpha_{ij}||y_j|\Big\} \le \|y\|_\infty\frac{1}{|\alpha_{ii}|}\sum_{\substack{j=1\\ j\neq i}}^{n}|\alpha_{ij}| \le C\|y\|_\infty.$$

Thus, (3.28) holds for all $1 \le i \le n$, and it may be written as

$$\forall y \in \mathbb{R}^n \qquad \|u\|_\infty = \|G_{GS}y\|_\infty \le C\|y\|_\infty.$$

Therefore, we have the required estimate $\|G_{GS}\|_\infty \le C \le 1$, and so the proof of (ii) is completed. □

Remark 3.1. *For a symmetric and positive definite matrix $A \in \mathbb{R}^{n,n}$ the Gauss-Seidel method converges, while, in general, Jacobi's method does not*

With the purpose of finding a method which converges faster than either Jacobi or the Gauss–Seidel method , speed-up techniques have been devised. One of these techniques is the *Successive Over-Relaxation (SOR) method* which was invented along with the advent of other sparse system techniques. Let $x^{(m)}$ be a known approximation of x and let $\tilde{x}^{(m+1)}$ be the approximation computed by the Gauss-Seidel method. It is expected that the linear combination

$$x^{(m+1)} = \omega\tilde{x}^{(m+1)} + (1-\omega)x^{(m)} \tag{3.30}$$

will have smaller error than $\tilde{x}^{(m+1)}$ under specific assumptions and for appropriate values of the relaxation parameter $\omega \neq 0$.

Consider the system $Ax = b$, where $\alpha_{ii} \neq 0$. For the SOR method we have

$$x_i^{(m+1)} = \frac{\omega}{\alpha_{ii}}\left[b_i - \sum_{j=1}^{i-1}\alpha_{ij}x_j^{(m+1)} - \sum_{j=i}^{n}\alpha_{ij}x_j^{(m)}\right] + (1-\omega)x_i^{(m)}, \quad 1 \le i \le n \tag{3.31}$$

or in matrix form

$$(D+\omega L)x^{(m+1)} = [(1-\omega)D - U]x^{(m)} + \omega b. \tag{3.32}$$

Obviously, the matrix $(D+\omega L)$ is invertible if $\alpha_{ii} \neq 0$ for all $1 \le i \le n$. If $\omega = 1$, then the SOR method coincides with the Gauss-Seidel method. It can be proved that if $A \in \mathbb{R}^{n,n}$ is symmetric and positive definite and $0 < \omega < 2$, then the SOR method converges. A natural question arises: Are there any values of $\omega \neq 1$ such that SOR converges faster than Gauss-Seidel method? The answer depends, of course, on finding a parameter ω which minimizes the spectral radius $\rho(G_\omega)$ of the iteration matrix $G_\omega = M_\omega^{(-1)}N_\omega$.

3.2 Linear iterative schemes

As shown in previous section, Jacobi's and Gauss-Seidel's methods are expressed by iterative schemes. That is to say, the estimation of x_{k+1} depends on known data like the matrix A, the x_k and the vector b. In general, iterative schemes are defined as follows:

Definition 3.3. *Let $A \in \mathbb{C}^{n\times n}$ with $\det(A) \neq 0$ and $f \in \mathbb{C}^n$. An **iterative scheme** to find an approximate solution to $Ax = f$ is a process to generate a sequence of approximations $\{x_k\}_{k=1}^{\infty}$ via an iteration of the form*

$$x_k = \phi(A, f, x_{k-1}, \ldots, x_{k-r}), \tag{3.33}$$

given the starting values $x_0, \cdots, x_{r-1} \in \mathbb{C}^n$. Here

$$\phi(\cdot,\cdot,\cdots,\cdot) : \mathbb{C}^{n\times n} \times \mathbb{C}^n \times \cdots \times \mathbb{C}^n \longrightarrow \mathbb{C}^n \tag{3.34}$$

*is called the **iteration function**. In case of $r = 1$ the process is a **two-layer scheme**, otherwase it is a **multilayer scheme**.*

Definition 3.4. *Let $A \in \mathbb{C}^{n\times n}$ with $\det(A) \neq 0$ and $f \in \mathbb{C}^n$. Set $x = A^{-1}f$. The two-layer iterative scheme*

$$x_k = \phi(A, f, x_{k-1}), \tag{3.35}$$

*is said to be **consistent** if and only if $x = \phi(A, f, x)$, i.e., $x = A^{-1}f$ is a fixed point of $\phi(A, f, \cdot)$. The scheme is **linear** if and only if*

$$\phi(A, \alpha f_1 + \beta f_2, \alpha x_1 + \beta x_2) = \alpha\phi(A, f_1, x_1) + \beta\phi(A, f_2, x_2), \tag{3.36}$$

$\forall \alpha, \beta \in \mathbb{C}$ and $x_1, x_2 \in \mathbb{C}^n$.

Proposition 3.3. *Let $A \in \mathbb{C}^{n\times n}$ with $\det(A) \neq 0$ and $f \in \mathbb{C}^n$. Any two-layer, linear, and consistent scheme can be written in the form*

$$x_{k+1} = x_k + Cr(x_k) = x_k + C(f - Ax_k), \tag{3.37}$$

for some matrix $C \in \mathbb{C}^{n\times n}$, where $r(z) = f - Az$ is the residual vector.

Proof. A two layer scheme is defined by an iteration function

$$\phi(\cdot,\cdot,\cdot) : \mathbb{C}^{n\times n} \times \mathbb{C}^n \times \mathbb{C}^n \longrightarrow \mathbb{C}^n.$$

Given ϕ, define the operator

$$Cz = \phi(A, z, \mathbf{0}).$$

This is a linear operator, due to the assumed linearity of the iteration function. Consequently, C can be identified as a square matrix. It follows from this definition, using the consistency and linearity of ϕ, that

$$\begin{aligned}(I_n - CA)w &= w - \phi(A, Aw, \mathbf{0}) \\ &= \phi(A, Aw, w) - \phi(A, Aw, \mathbf{0}) \\ &= \phi(A, \mathbf{0}, w).\end{aligned}$$

Furthermore by linearity, we can write

$$\begin{aligned}x_{k+1} &= \phi(A, f + \mathbf{0}, 0) + \phi(A, \mathbf{0}, x_k) \\ &= Cf + (I_n - CA)x_k \\ &= x_k + C(f - Ax_k),\end{aligned}$$

as we intended to show. □

On the grounds that the equation (3.37) can be written as

$$C^{-1}(x_{k+1}-x_k)+Ax_k=f.$$

where C is invertible, we have the following definitions.

Definition 3.5. *Let $A\in\mathbb{C}^{n\times n}$ with $\det(A)\neq 0$ and $f\in\mathbb{C}^n$. A scheme of the form*

$$B_{k+1}(x_{k+1}-x_k)+Ax_k=f, \tag{3.38}$$

*where $B_{k+1}\in\mathbb{C}^{n\times n}$ is invertible is called an **adaptive two-layer scheme.** If $B_{k+1}=B$, where B is invertible and independent of k, then the scheme is called **stationary two-layer scheme** and the matrix **B** is called the **iterator**. If $B_{k+1}=\frac{1}{\alpha_{k+1}}I_n$, where $\alpha_{k+1}\in\mathbb{C}_*$, then we say that the adaptive two-layer scheme is **explicit**.*

With regard to stationary two-layer schemes, we assume that B is invertible so that (3.38) becomes

$$x_{k+1}=x_k+B^{-1}(f-Ax_k). \tag{3.39}$$

The last equality shows that if $\{x_k\}$ converges, then it must converge to $x=A^{-1}f$.

Definition 3.6. *Let $A\in\mathbb{C}^{n\times n}$ be invertible and $f\in\mathbb{C}^n$. Suppose that $x=A^{-1}f$ and consider the stationary two-layer scheme (3.39) defined by the invertible matrix $B\in\mathbb{C}^{n\times n}$. The matrix $T=I_n-B^{-1}A$ is called the **error transfer matrix** and satisfies*

$$e_{k+1}=Te_k,$$

*where $e_k=x-x_k$ is the **error** at step k.*

Theorem 3.2. *Let $A\in\mathbb{C}^{n\times n}$. The following are equivalent.*

1. *A is convergent to zero.*
2. *For some induced matrix norm*
$$\lim_{k\longrightarrow\infty}\|A^k\|=0.$$
3. *For all induced matrix norms*
$$\lim_{k\longrightarrow\infty}\|A^k\|=0.$$
4. $\rho(A)<1$.
5. *For all $x\in\mathbb{C}^n$,*
$$\lim_{k\longrightarrow\infty}A^kx=0$$

Before proving the above theorem, we establish a useful inequality:

Proposition 3.4. *For every matrix $A\in\mathbb{C}^{n\times n}$ is true that*

$$\|A\|_{max}\leq\|A\|_\infty\leq n\|A\|_\infty$$

Proof. For a matrix $A\in\mathbb{C}^{n\times n}$, $\|\cdot\|_{max}$ and $\|\cdot\|_\infty$ norms define as

$$\|A\|_{max}=\max_{1\leq i,j\leq n}|\alpha_{ij}|$$

and

$$\|A\|_\infty=\max_{1\leq i\leq n}\sum_{j=1}^{n}|\alpha_{ij}|$$

The left part of the inequality is obvious. We will prove the right part of the inequality.

$$\|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |\alpha_{ij}| \le \max_{1\le i\le n} \sum_{j=1}^{n} \max_{1\le j\le n} |\alpha_{ij}| = n \max_{1\le i,j\le n} |\alpha_{ij}| = n\|A\|_\infty,$$

which completes the proof. □

Now we go back to proving the theorem:

Proof. <u>$1 \Rightarrow 2$:</u> Since A is convergent to zero, using (3.4),

$$\frac{1}{n}\|A^k\|_\infty \le \|A^k\|_{max} \to 0.$$

Thus $\|A^k\|_\infty \to 0$.

<u>$2 \Rightarrow 1$:</u> Suppose that $\lim_{k\to\infty} \|A^k\| = 0$, for some induced matrix norm. Since all matrix norms are equivalent,

$$\lim_{k\to\infty} \|A\|_\infty = 0.$$

Using (3.4),

$$\|A^k\|_{max} \le \|A^k\|_\infty \to 0.$$

Therefore, A is convergent to zero.

<u>$2 \Rightarrow 4$:</u> From the Schur factorization theorem we have that $A = UTU^H$, where T is upper triangular and U is unitary. Then

$$A^k = UT^kU^H.$$

Also, it's true that $\rho(A) \le \|A\|$, for any induced matrix norm and if λ is an eigenvalue of A, then λ^k is an eigenvalue of A^k. Therefore,

$$0 \le \rho^k(A) = \rho(A^k) \le \|A^k\|.$$

Thus, if $\|A^k\| \to 0$, it follows that

$$\rho^k(A) \to 0.$$

This implies $\rho(A) < 1$.

<u>$4 \Rightarrow 2$:</u> By the lemma 3.1 there is an induced matrix norm $\|\cdot\|_\star$ such that

$$\|A\|_\star \le \rho(A) + \varepsilon,$$

for any $\varepsilon > 0$. Recall that the choice of $\|\cdot\|_\star$ depends upon A and $\varepsilon > 0$. Since, by assumption $\rho(A) < 1$, there is an $\varepsilon > 0$ and an induced norm $\|\cdot\|_\star$, such that

$$\|A^k\|_\star \le \rho(A) + \varepsilon = \theta < 1.$$

Then, using sub-multiplicativity,

$$\|A^k\|_\star \le \|A\|_\star^k \le \theta^k \to 0.$$

Consequently,

$$\lim_{k\to\infty} \|A^k\|_\star = 0.$$

<u>$2 \Rightarrow 3$:</u> This follows from the equivalence of norms. If convergence is observed in one induced norm, it is observed in all induced norms.

<u>$3 \Rightarrow 5$:</u> Suppose that $\lim_{k\to\infty} \|A^k\| = 0$ for all induced matrix norms. Let $x \in \mathbb{C}^n$ be arbitrary. Then,

$$\|A^k x\|_\infty \leq \|A^k\|_\infty \|x\|_\infty \to 0$$

Since $\|A^k\|_\infty \to 0$. Hence $\|A^k x\|_\infty \to 0$. This implies

$$\lim_{k\to\infty} A^k x = 0.$$

<u>$5 \Rightarrow 1$:</u> Suppose that, for any $x \in \mathbb{C}^n$,

$$\lim_{k\to\infty} A^k x = 0.$$

Then, it follows that, $\forall x, y \in \mathbb{C}^n$,

$$y^H A^k x \to 0.$$

Now, suppose $y = e_i$ and $x = e_j$. Since

$$y^H A^k x = e_i^H A^k e_j = [A^k]_{i,j},$$

then it follows that

$$\lim_{k\to\infty} [A^k]_{i,j} = 0.$$

This implies that

$$\lim_{k\to\infty} \|A^k\|_{max} = 0.$$

Hence, A is convergent to zero. □

At this point it is useful to state a theorem about the convergence of linear schemes.

Theorem 3.3. *Suppose that $A, B \in \mathbb{C}^{n\times n}$ are invertible, $f, x_0 \in \mathbb{C}^n$ given and $x = A^{-1} f$.*

1. *The sequence $\{x_k\}_{k=1}^\infty$ defined by the linear, two-layer, stationary iterative scheme (3.39) converges to x for any starting point x_0 iff $\rho(T) < 1$, where T is the error transfer matrix $T = I_n - B^{-1}A$.*
2. *A sufficient condition for the convergence of $\{x_k\}_{k=1}^\infty$ for any starting point x_0 is the condition that $\|T\| < 1$, for some induced matrix norm.*

Proof. It's true that

$$e_k = Te_{k-1} = T^2 e_{k-2} = \cdots = T^k e_0. \tag{3.40}$$

Observe that

$$x_k \xrightarrow{k\longrightarrow\infty} x = A^{-1} f \quad \text{iff} \quad e_k \xrightarrow{k\longrightarrow\infty} 0$$

Suppose that $x_k \longrightarrow x = A^{-1} f$, as $k \longrightarrow \infty$, for any x_0. Then $e_k \xrightarrow{k\longrightarrow\infty} 0$ for any e_0. Set $e_0 = w$, where (λ, w) is any eigenpair of T, with $\|w\|_\infty = 1$. Then

$$e_k = \lambda^k e_0,$$

and

$$|\lambda|^k = |\lambda|^k \|w\|_\infty = \|e_k\|_\infty \to 0.$$

It follows that $|\lambda| < 1$. Since λ was arbitrary, $\rho(T) < 1$. If $\rho(T) < 1$, appealing to Theorem 3.2,

$$\lim_{k\to\infty} e_k = \lim_{k\to\infty} T^k e_0 = 0,$$

for any e_0. Hence, $x_k \to x = A^{-1} f$, as $l \to \infty$, for any x_0. Suppose now that $\|T\| < 1$ for some induced matrix norm. Since, for any induced matrix norm,

$$\rho(T) \leq \|T\|,$$

it follows that $\rho(T) < 1$. Again, by Theorem 3.2,

$$\lim_{k\to\infty} e_k = \lim_{k\to\infty} T^k e_0 = 0,$$

for any e_0. □

Theorem 3.4. *Let $A, B \in \mathbb{C}^{n\times n}$ be invertible, $x_0, f \in \mathbb{C}^n$ and $x = A^{-1}f$. Let $x_{k\,k=1}^{\infty}$ be the sequence generated by the linear, two-layer, stationary scheme (3.39). The following estimates hold*

$$\|x - x_k\| \le \|T\|^k \|x - x_0\|,$$

$$\|x - x_k\| \le \frac{\|T\|^k}{1 - \|T\|} \|x_1 - x_0\|.$$

Proof. It follows that $e_k = T^k e_0$. By using the consistency and sub-multiplicativity of the induced matrix norm, we find

$$\|e_k\| \le \|T^k\| \|e_0\| \le \|T\|^k \|e_0\|,$$

which proves the first estimate.
For the second one, observe that $e_k = T^{k-1} e_1$, and thus $Te_k = T^k e_1$. Substracting the last expression from $e_k = T^k e_0$, we find

$$(I_n - T)e_k = T^k (x_1 - x_0).$$

We have that $\|T\| < 1$. Using the reverse triangle inequality for any $x \in \mathbb{C}^n$,

$$\|(I_n - T)x\| \ge |\|x\| - \|Tx\|| \ge (1 - \|T\|)\|x\|.$$

The inequality implies that, if $(I_n - T)x = 0$ then $x = 0$. Therefore, $I_n - T$ is invertible.
To obtain the norm estimate, notice that

$$1 = \|I_n\| = \|(I_n - T)(I_n - T)^{-1}\| = \|(I_n - T)^{-1} - T(I_n - T)^{-1}\| \ge \|(I_n - T)^{-1}\| - \|T\| \|(I_n - T)^{-1}\|,$$

thus

$$\|(I_n - T)^{-1}\| \le \frac{1}{1 - \|T\|}.$$

Hence,

$$e_k = (I_n - T)^{-1} T^k (x_1 - x_0)$$

and using the consistency and sub-multiplicativity of the norm, we get

$$\|e_k\| \le \|(I_n - T)^{-1}\| \|T\|^k \|x_1 - x_0\| \le \frac{1}{1 - \|T\|} \|T\|^k \|x_1 - x_0\|.$$

□

3.3 Matrix splitting methods

As we describe in subsection 3.1.2, a matrix A can be split as

$$A = L + D + U.$$

For a stationary two-layer method, if we choose $B = D$ as iterator we would have the *Jacobi method* and error transfer matrix

$$T = T_J = I_n - D^{-1}A = -\begin{pmatrix} 0 & \frac{a_{1,2}}{\alpha_{1,1}} & \cdots & \frac{a_{1,n}}{\alpha_{1,1}} \\ \frac{a_{2,1}}{\alpha_{2,2}} & 0 & \cdots & \frac{a_{2,n}}{\alpha_{2,2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n,1}}{\alpha_{n,n}} & \cdots & \frac{a_{n,n-1}}{\alpha_{n,n}} & 0 \end{pmatrix} \tag{3.41}$$

Consider the system $Ax = f$. According to (3.2) and the matrix T_J, the estimating formula of $x_i^{(k+1)}$ can be written in the form

$$x_i^{(k+1)} = -\frac{1}{\alpha_{i,i}} \sum_{\substack{j \neq i \\ j=1}}^{n} \alpha_{i,j} x_j^k + \frac{1}{\alpha_{i,i}} f_i = [T_J x^k]_i + [D^{-1} f]_i$$

In the same way, the Gauss-Seidel method defined by (3.3) with error transfer matrix

$$T_{GS} = -(L+D)^{-1}U = -(A-U)^{-1}U. \tag{3.42}$$

We prove a theorem which gives us an important conclusion about the relation between the convergence of Jacobi method and the convergence of Gauss-Seidel method.

Theorem 3.5. *Let $A \in \mathbb{C}^{n\times n}$ be tridiagonal with non zero diagonal elements. Denote by T_J and T_{GS} the error transfer matrices of the Jacobi and Gauss-Seidel methods, respectively. Then we have*

$$\rho(T_{GS}) = \rho(T_J)^2.$$

In particular, one method converges iff the other method does.

Proof. Suppose that

$$A = \begin{pmatrix} b_1 & c_2 & 0 & \cdots & 0 \\ a_2 & b_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & c_{n-1} & 0 \\ \vdots & \ddots & a_{n-1} & b_{n-1} & c_n \\ 0 & \cdots & 0 & a_n & b_n \end{pmatrix}$$

where $b_i \neq 0$, for all $1 \leq i \leq n$. Let $0 \neq \mu \in \mathbb{C}$ and define

$$M(\mu) = DAD^{-1},$$

where $D = diag[\mu, \mu^2, \cdots, \mu^n]$. Then

$$M(\mu) = \begin{pmatrix} b_1 & \mu^{-1}c_2 & 0 & \cdots & 0 \\ \mu a_2 & b_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \mu^{-1}c_{n-1} & 0 \\ \vdots & \ddots & \mu a_{n-1} & b_{n-1} & \mu^{-1}c_n \\ 0 & \cdots & 0 & \mu a_n & b_n \end{pmatrix}$$

It's true that $\det(M(\mu)) = \det(M(1)) = \det(A)$. Suppose the matrix $Q(\mu) = diag(\mu, \mu^2, \cdots, \mu^n)$ with the property: $Q(\mu)Q^{-1}(\mu) = I$ and $Q^{-1}(\mu) = Q(\mu^{-1})$. In addition we have the equality

$$\begin{aligned} M(\mu)Q(\mu) = Q(\mu)M(1) \quad &\Rightarrow \quad Q^{-1}M(\mu)Q(\mu) = M(1) \\ &\Rightarrow \quad det(Q^{-1})det(M(\mu))det(Q(\mu)) = det(M(1)) \\ &\Rightarrow \quad det(M(\mu)) = det(M(1)). \end{aligned}$$

Let $A = L+D+U$ where L, D, U define as usual. From (3.41) we have that $T_J = I_n - D^{-1}A$. Therefore, the eigenvalues of T_J are the zeros of the characteristic polynomial

$$\begin{aligned} p_J(\lambda) &= \det(T_J - \lambda I_n) \\ &= \det(-D^{-1}(L+U) - \lambda I_n) \\ &= \det[-D^{-1}(L+U+\lambda D)] \\ &= \det(-D^{-1})q_J(\lambda), \end{aligned}$$

where we defined the polynomial

$$q_J(\lambda) = det(L+\lambda D+U).$$

On the other hand, form (3.42) we have that $T_{GS} = -(L+D)^{-1}U$, and so its eigenvalues are the zeros of

$$p_{GS}(\lambda) = det(T_{GS} - \lambda I_n) = det(-(L+D)^{-1})q_{GS}(\lambda),$$

where

$$q_{GS}(\lambda) = \det(\lambda L+\lambda D+U).$$

Notice now that $0 = q_{GS}(0) = q_J(0)$. In addition, both matrices involved in the definitions of q_J and q_{GS}, respectively, are tridiagonal. By the previous statement about determinants of tridiagonal matrices we have that, if $\lambda \neq 0$,

$$\begin{aligned} q_{GS}(\lambda^2) &= det(\lambda^2 L+\lambda^2 D+U) \\ &= \lambda^n det(\lambda L+\lambda D+\lambda^{-1}U) \\ &= \lambda^n det(L+\lambda D+U) \\ &= \lambda^n q_J(\lambda) \end{aligned}$$

and so this holds for all $\lambda \in \mathbb{C}$. The previous reltion shows that

$$\lambda \in \sigma(T_{GS}) \Rightarrow \lambda^{\frac{1}{2}}, \quad -\lambda^{\frac{1}{2}} \in \sigma(T_J)$$

and

$$(\lambda \in \sigma(T_J) \Longleftrightarrow -\lambda \in \sigma(T_J)) \Rightarrow \lambda^2 \in \sigma(T_{GS}).$$

□

In previous section we referred to SOR method which define via (3.32). In this case the error transfer matrix is given by

$$T_\omega = (L+\omega^{-1}D)^{-1}((\omega^{-1}-1)D-U)$$

and the iterator is

$$B_\omega = L+\omega^{-1}D.$$

It's necessary to choose a valid value for ω to achieve convergence. A necessary condition for this is given by the theorem below.

Theorem 3.6. *Let $A \in \mathbb{C}^{n\times n}$ have non zero diagonal entries. A necessary condition for convergence of the relaxation method is that $\omega \in (0,2)$.*

Proof. Since $A \in \mathbb{C}^{n\times n}$ has non zero diagonal entries, the relaxation method is well defined. A necessary and sufficient condition for convergence is $\rho(T_\omega) < 1$. Eigenvalues are roots of the characteristic polynomial,

$$\chi_T(\lambda) = \det(T_\omega - \lambda I_n) = (-1)^n \prod_{i=1}^{n}(\lambda - \lambda_i).$$

If $\lambda = 0$ then

$$\chi_T(0) = det(T_\omega) = \prod_{i=1}^{n}(\lambda_i).$$

However, if $|\det(T_\omega)| \geq 1$ then there must be at least one eigenvalue that satisfies $|\lambda_i| \geq 1$, so the method cannot converge. Thus,

$$|\det(T_\omega) < 1.$$

□

In previous sections we discussed the convergence properties of the Jacobi and Gauss–Seidel (GS) methods and we saw that GS converges faster than the Jacobi method. However, the results shown above indicate that between these three methods, SOR is faster, given a suitable relaxation parameter ω.

In the example below we compare the convergence of GS and SOR methods for a specific matrix, the 48×48 matrix "bcsstk01" from the Suite Sparse Matrix Collection:

Example 3.3.1. *Suppose the system $Ax = b$ where A is the 48×48 matrix and b is the unit vector. If we try to approach the solution x with Jacobi's method, then the result would be that the method did not converges in 4000 iterations.*
On the other hand, GS and SOR converge in less than 4000 iterations. Suppose that the relaxation parameter is $\omega = 1.8$, the number of max iterations is 4000 and estimate error is $e = 10^{-4}$. Running the code for GS and SOR methods we have the results below:

```
Gauss-Seidel converged
Number of iterations =  3462. || b - A x || = 0.000100289
Times: 136.97223006700005 seconds

SOR converged
Number of iterations =   437. || b - A x || = 9.90254e-05
Times: 16.83217748299967 seconds
```

Figure 3.1: Results for GS and SOR methods

Observe that SOR converges faster with smaller residual error. This conclusion can be more obvious by sketching the comparison plot between residual error and iteration as it shows below:

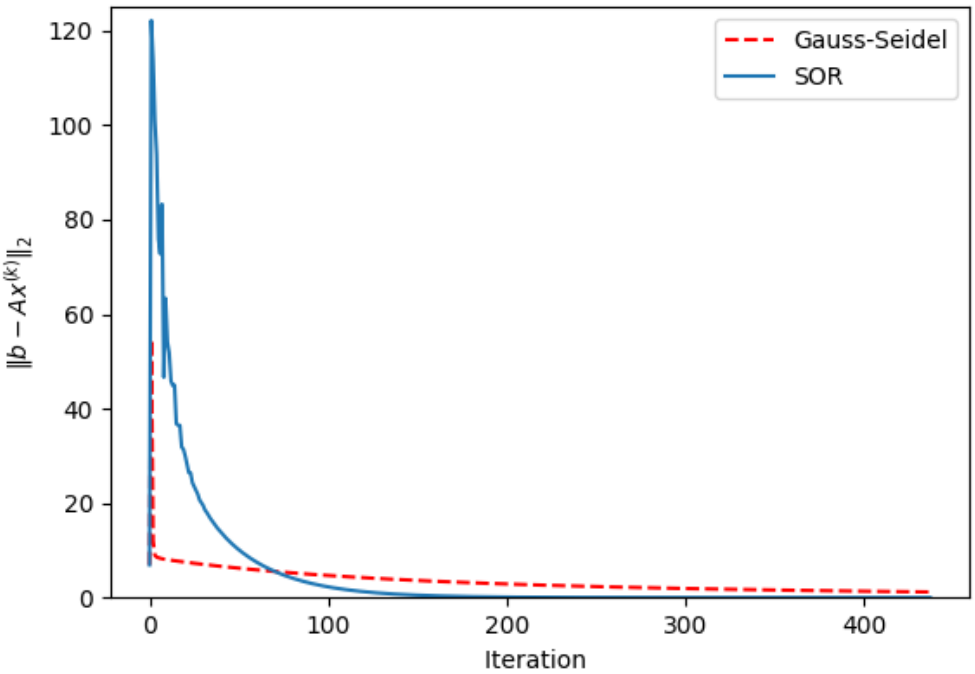

Figure 3.2: Comparison plot between residual error and iteration

CHAPTER 4

Gradient methods and Krylov subspaces

4.1 Introductory concepts

Suppose we have the system $Ax = f$, where $A \in \mathbb{C}^{n\times n}$ is Hermitian positive definite (HPD). The following theorem summarizes some ot the properties of HPD matrices.

Theorem 4.1. *Suppose that $A \in \mathbb{C}^{n\times n}$ is Hermitian positive definite. Then the following are true:*

1. *The expression*
$$(x,y)_A = (Ax,y)_2 = y^H Ax, \qquad \forall x,y \in \mathbb{C}^n, \tag{4.1}$$
defines an inner product on $\mathbb{C}^n$.

2. *The object $\|x\|_A = \sqrt{x^H Ax}$, where $x \in \mathbb{C}^n$, defines a norm on $\mathbb{C}^n$.*

3. *Let the eigenvalues of A be ordered so that $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Then*
$$\sqrt{\lambda_1}\|x\|_2 \leq \|x\|_A \leq \sqrt{\lambda_n}\|x\|_2,$$
for any $x \in \mathbb{C}^n$.

4. *Let $f \in \mathbb{C}^n$ be given. Then $x = A^{-1}f$ if and only if x minimizes the quadratic functional $E_A : \mathbb{C}^n \to \mathbb{R}$, defined by*
$$E_A(z) = \frac{1}{2}z^H Az - \Re(z^H f).$$

Proof. We prove each assertion of the theorem separately:

1. For all $x \in \mathbb{C}^n$, it's true that
$$(x,x)_A = (Ax,x)_2 = x^H Ax > 0,$$
because A is HPD, which implies that $x^H Ax > 0, \quad \forall x \in \mathbb{C}^n, x \neq 0$.
Now, for $x,y \in \mathbb{C}^n$,
$$\begin{aligned}(x,y)_A &= (Ax,y)_2 = y^H Ax = y^H A^H x = (Ay)^H x \\ &= \overline{x^H Ay} \\ &= \overline{(Ay,x)_2} \\ &= \overline{(y,x)_A},\end{aligned}$$

which establishes the symmetry of the inner product.
Finally, for $x, y, w \in \mathbb{C}^n$ and any $\lambda_1, \lambda_2 \in \mathbb{C}$, we have

$$\begin{aligned}
(\lambda_1 x + \lambda_2 y, w)_A &= (A(\lambda_1 x + \lambda_2 y), w)_2 \\
&= w^H A(\lambda_1 x + \lambda_2 y) \\
&= w^H A\lambda_1 x + w^H A\lambda_2 y \\
&= \lambda_1 (Ax, w)_2 + \lambda_2 (Ay, w)_2 \\
&= \lambda_1 (x, w)_A + \lambda_2 (y, w)_A.
\end{aligned}$$

Thus, (4.1) defines an inner product on $\mathbb{C}^n$.

2. Observe that

$$\|x\|_A^2 = (x, x)_A.$$

The fact that $(\cdot,\cdot)_A$ is an inner product implies the positivity of $\|\cdot\|_A$ and the fact that $\|\lambda x\|_A = |\lambda|\,\|x\|_A$, for $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^n$. We will prove the triangle inequailty, i.e. the inequality

$$\|x + y\|_A \leq \|x\|_A + \|y\|_A.$$

First of all, we going to prove Cauchy inequality for the above inner product by calculating the inner product of the vector $x - \lambda y$ with itself.

$$\begin{aligned}
(x - \lambda y, x - \lambda y)_A &= (x, x)_A - (x, \lambda y)_A - (\lambda y, x)_A + (\lambda y, \lambda y)_A \\
&= (x, x)_A - \overline{\lambda}(x, y)_A - \lambda (y, x)_A + |\lambda|^2 (y, y)_A
\end{aligned}$$

For $\lambda = \frac{(x,y)_A}{(y,y)_A}$:

$$\begin{aligned}
0 \leq (x - \lambda y, x - \lambda y)_A &= (x, x)_A - \frac{(y, x)_A}{(y, y)_A}(x, y)_A - \frac{(x, y)_A}{(y, y)_A}(y, x)_A + \frac{|(x, y)_A|^2}{|(y, y)_A|^2}(y, y)_A \\
&= (x, x)_A - 2\frac{(x, y)_A (y, x)_A}{(y, y)_A} + \frac{|(x, y)_A|^2}{(y, y)_A} \\
&= (x, x)_A - \frac{|(x, y)_A|^2}{(y, y)_A}
\end{aligned}$$

from which it follows that

$$|(x, y)_A|^2 \leq (x, x)_A (y, y)_A = \|x\|_A^2 \|y\|_A^2. \tag{4.2}$$

From (4.2) it follows that

$$\begin{aligned}
\|x + y\|_A^2 &= (x + y, x + y)_A \\
&= (x, x)_A + (x, y)_A + (y, x)_A + (y, y)_A \\
&= \|x\|_A^2 + \|y\|_A^2 + (x, y)_A + (y, x)_A.
\end{aligned} \tag{4.3}$$

However,

$$\begin{aligned}
(x, y)_A = (Ax, y)_2 &= (Ax)^H y \\
&= x^H A^H y \\
&= x^H A y \\
&= (Ay, x)_2 \\
&= (y, x)_A
\end{aligned}$$

Thus, using the symmetry of the inner prodcut $(\cdot,\cdot)_A$ and the Cauchy–Schwarz inequality (4.2), relation (4.3) becomes

$$\begin{aligned}\|x+y\|_A^2 &= \|x\|_A^2+\|y\|_A^2+2(x,y)_A\\ &\le \|x\|_A^2+\|y\|_A^2+2\|x\|_A^2\|y\|_A^2=(\|x\|_A+\|y\|_A)^2\end{aligned}$$

This establishes that $\|\cdot\|_A$ is a norm.

3. Suppose $S=\{w_1,\dots,w_n\}$ is an orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of A (i.e. $Aw_i=\lambda_i w_i$ and $(w_i,w_j)=w_i^H w_j=0$ for $i\neq j$). Let $x\in\mathbb{C}^n$ be arbitrary, and write $x=\sum_{i=1}^n c_i w_i$, for some for uniquely determined constants $c_i\in\mathbb{C}$, $i=1,\dots,n$. Then

$$\begin{aligned}\|x\|_A^2=x^HAx=x^H\sum_{i=1}^n Ac_iw_i &= x^H\sum_{i=1}^n c_i\lambda_i w_i\\ &= \sum_{i=1}^n c_i\lambda_i(x^Hw_i)\\ &= \sum_{i=1}^n c_i\lambda_i(\sum_{j=1}^n \bar{c}_j w_j^H w_i)\\ &= \sum_{i=1}^n c_i\lambda_i(\sum_{j=1}^n \bar{c}_j\delta_{ij})\\ &= \sum_{i=1}^n c_i\bar{c}_i\lambda_i\\ &= \sum_{i=1}^n |c|^2\lambda_i \qquad (4.4)\\ &\le \lambda_n\sum_{i=1}^n |c|^2=\lambda_n\|x\|_2^2.\end{aligned}$$

Thus, $\|x\|_A\le\sqrt{\lambda_n}\|x\|_2$. For the left part of inequality, it's obvious that we can take the lower bound of (4.4)

$$\|x\|_A^2\ge\lambda_1\sum_{i=1}^n|c|^2=\lambda_1\|x\|_2^2,$$

and we have the needed $\|x\|_A\ge\sqrt{\lambda_1}\|x\|_2$.

4. Suppose that $x=A^{-1}f$. Let $y\in\mathbb{C}^n$ be arbitrary and consider

$$\begin{aligned}E_A(x+y) &= \frac{1}{2}(x+y)^HA(x+y)-\Re(x^Hf)-\Re(y^Hf) \qquad (4.5)\\ &= \frac{1}{2}x^HAx+\frac{1}{2}y^HAy+\frac{1}{2}x^HAy+\frac{1}{2}y^HAx-\Re(x^Hf)-\Re(y^Hf).\end{aligned}$$

Note that $\Re(A)=\frac{1}{2}(A+A^H)$, so that $\Re(y^HAx)=\frac{1}{2}(y^HAx+(y^HAx)^H)$, and $x^HAy=x^HA^Hy=(Ax)^Hy=(y^HAx)^H$. Then

$$\begin{aligned}E_A(x+y) &= \frac{1}{2}x^HAx+\frac{1}{2}y^HAy+\frac{1}{2}(y^HAx+(y^HAx)^H)-\Re(x^Hf)-\Re(y^Hf)\\ &= \frac{1}{2}x^HAx+\frac{1}{2}y^HAy+\Re(y^HAx)-\Re(x^Hf)-\Re(y^Hf)\\ &= E_A(x)+\frac{1}{2}y^HAy+\Re(y^H(Ax-f))\\ &= E_A(x)+\frac{1}{2}y^HAy\ge E_A(x).\end{aligned}$$

The last inequality holds because A is HPD so that $y^H Ay$ is non-negative. Equality holds if and only if $y = 0$, therefore x minimizes E_A.
Conversely, suppose that x minimizes the functional E_A, let $u \in \mathbb{C}^n$ be an arbitrary unit vector, and define $g(s,t) = E_A(x+\alpha u)$, where $\alpha = s+it$, $s,t \in \mathbb{R}$. Then

$$\begin{aligned} g(s,t) &= \frac{1}{2}(x+\alpha u)^H A(x+\alpha u) - \Re((x+\alpha u)^H f) \\ &= \frac{1}{2}x^H Ax + \frac{1}{2}x^H Aau + \frac{1}{2}\overline{\alpha}u^H Ax + \frac{1}{2}\overline{\alpha}u^H Aau - \Re(x^H f) - \Re(\overline{\alpha}u^H f) \end{aligned}$$

A simple calculation shows that (4.6) may be written as

$$\begin{aligned} g(s,t) &= \frac{1}{2}x^H Ax + \Re(\overline{\alpha}u^H Ax) + \frac{|\alpha|^2}{2}u^H Au - \Re(x^H f) - \Re(\overline{\alpha}u^H f) \\ &= E_A(x) + \Re(\overline{\alpha}u^H Ax) + \frac{|\alpha|^2}{2}u^H Au - \Re(\overline{\alpha}u^H f) \\ &= E_A(x) + \frac{|\alpha|^2}{2}u^H Au + \Re(\overline{\alpha}u^H(Ax-f)) \\ &= E_A(x) + \frac{|\alpha|^2}{2}u^H Au + \Re(\overline{\alpha})\Re(u^H(Ax-f)) - \Im(\overline{\alpha})\Im(u^H(Ax-f)) \\ &= E_A(x) + s\Re(u^H(Ax-f)) + t\Im(u^H(Ax-f)) + \frac{s^2+t^2}{2}u^H Au. \end{aligned}$$

Considering all the above, g is a strictly convex, quadratic function on $\mathbb{R}^2$. On the grounds that x minimizes g, we conclude that g has minimum at $(s,t) = (0,0)$. Hence,

$$0 = \frac{\partial g}{\partial s}(0,0) = \Re(u^H(Ax-f)),$$

and

$$0 = \frac{\partial g}{\partial t}(0,0) = \Im(u^H(Ax-f)),$$

where u is arbitrary. Therefore, it follows that $Ax = f$.

This concludes the proof of the the theorem. □

In Theorem 4.1 we showed that an HPD matrix defines an inner product. The converse is also true and this is established in the following proposition.

Proposition 4.1. *Suppose that $(\cdot,\cdot) : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}$ is an inner product. There exists a unique HPD matrix $A \in \mathbb{C}^{n\times n}$ such that*

$$(x,y) = (Ax,y)_2 = (x,y)_A, \qquad \forall x,y \in \mathbb{C}^n.$$

Definition 4.1. *Suppose that $(\cdot,\cdot) : \mathbb{C}^n \times \mathbb{C}^n \to \mathbb{C}$ is an inner product and $A \in \mathbb{C}^{n\times n}$ is its associated HPD matrix. We say that $B \in \mathbb{C}^{n\times n}$ is **self-adjoint** with respect to this inner product if and only if*

$$(x,By) = (x,By)_A = (Bx,y)_A = (Bx,y), \qquad \forall x,y \in \mathbb{C}^n.$$

*We say that B is **self-adjoint positive definite** with respect to this inner product if and only if B is self-adjoint and satisfies*

$$(x,Bx) = (x,Bx)_A > 0, \qquad \forall x \in \mathbb{C}^n_\star.$$

*We say that two vectors $x,y \in \mathbb{C}^n$ are **A-orthogonal** (or **A-conjugate**) if and only if*

$$(x,y) = (x,y)_A = 0.$$

We say that a set $S \subset \mathbb{C}^n$ of non-zero vectors is called ***A-orthogonal*** *(or* ***A-conjugate****) if and only if whenever $x, y \in S$, and $x \neq y$, then*

$$(x,y) = (x,y)_A = 0.$$

We say that $S \subset \mathbb{C}^n$ is ***A-orthonormal*** *iff S is A-orthogonal and*

$$\|x\|_A = 1, \quad \forall x \in S.$$

Theorem 4.2. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD and $B \in \mathbb{C}^{n\times n}$ is self-adjoint with respect to $(\cdot,\cdot)_A$. Then all of the eigenvalues of B are real and there is an A-orthonormal basis of $\mathbb{C}^n$ consisting of eigenvectors of B.*

Before proving this theorem, we will state two useful propositions:

Lemma 1. *Let $A \in \mathbb{C}^{n\times n}$. Then there exist matrices $U, D \in \mathbb{C}^{n\times n}$ with U unitary and D upper triangular, such that*

$$A = UDU^H.$$

Proposition 4.2. *Let $A \in \mathbb{C}^{n\times n}$ be self-adjoint (Hermitian). Then $\sigma(A) \subseteq \mathbb{R}$ and there is a unitary matrix $U \in \mathbb{C}^{n\times n}$ such that*

$$A = UDU^H,$$

where $D = diag[\lambda_1, \cdots, \lambda_n]$. Furthermore, there exists an orthonormal basis $B = \{u_1, \cdots, u_n\}$ of eigenvectors of A for the space $\mathbb{C}^n$, and $Au_i = \lambda_i u_i$, $i = 1, \cdots, n$.

Now we return to the proof of Theorem 4.2

Proof. Consider the matrix $C = L^H B L^{-H}$, where L is the lower triangular matrix of the Cholesky decomposition of the matrix A, so that $A = LL^H$. We have $L^H = L^{-1}A$ and $L = AL^{-H}$. Since B is self-adjoint with respect to $(\cdot,\cdot)_A$ we have that $B^H A = AB$. We will show that C is a Hermitian matrix. Indeed,

$$\begin{aligned} C = L^H B L^{-H} &= L^{-1} A B L^{-H} \\ &= L^{-1} B^H A L^{-H} \\ &= L^{-1} B^H L \\ &= C^H. \end{aligned}$$

Applying Proposition 4.2, matrix C is equal to

$$L^H B L^{-H} = C = UDU^{-1},$$

where $U \in \mathbb{C}^{n\times n}$ and D is a diagonal matrix with entries $D = \mathrm{diag}[\lambda_1, \ldots, \lambda_n]$. Hence, B is similar to a diagonal matrix with real entries

$$B = (L^{-H}U)D(L^{-H}U)^{-1}.$$

Setting $M = L^{-H}U$, we have

$$BM = MD$$

which implies that the columns of the invertible matrix M are eigenvectors of B.

Let us now show that the columns of M form an A-orthonormal set. The columns of M are eigenvectors of B so we know that they are A-orthogonal. We have to establish that $\|x\|_A = 1$, for any eigenvector x of B. First, we know that the matrix L is an unitary matrix, i.e. $L^{-1} = L^H$. As a consequence, we have that

$$M^{-1} = (L^{-H}U)^{-1} = U^{-1}L^H = U^H L^H = U^H L^{-1} = (L^{-H}U)^H = M^H.$$

By definition, if the invertible matrix is unitary, then its columns form an orthonormal set. Moreover it is true that

$$x_i^H x_i = x_i^H L L^{-1} x_i = x_i^H L L^H x_i = x_i^H A x_i = (x_i, x_i)_A = \|x_i\|_A^2 = 1,$$

so the columns of M form an A–orthonormal set. □

4.2 Gradient descent methods

Gradient descent methods are frequently used for the minimization of nonlinear functions. The general idea of the method is to proceed along a direction of descent, computing at each step an appropriate step size. The difficulty, of course, lies in detemining the steps size. In that respect, we will make use of the following definitions:

Definition 4.2. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n$, and define the quadratic functional $E_A : \mathbb{C}^n \to \mathbb{R}$ via*

$$E_A(z) = \frac{1}{2} z^H A z - \Re(z^H f).$$

*A **gradient descent method** is a two-layer iterative scheme to approximate $x = A^{-1} f$. Starting from an arbitrary initial guess x_0, the iterations proceed as*

$$x_k = x_{k-1} + \alpha_k d_{k-1}, \qquad k = 1, 2, 3, \ldots.$$

*where $d_{k-1} \in \mathbb{C}^n$ is the $(k-1)$-st **search direction**, supplied by the algorithm, and $\alpha_k \in \mathbb{R}$ is the **step size** given by the condition*

$$\alpha_k = \arg\min_{\alpha\in\mathbb{C}} E_A(x_{k-1} + \alpha d_{k-1}),$$

*which is called a **line search.***

The next theorem gives a formula to calculate α_k at each iteration step:

Theorem 4.3. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n$, and define the quadratic functional $E_A : \mathbb{C}^n \to \mathbb{R}$ via*

$$E_A(z) = \frac{1}{2} z^H A z - \Re(z^H f).$$

Suppose that the search direction $d_{k-1} \in \mathbb{C}^n_\star$, and previous iterate $x_{k-1} \in \mathbb{C}^n$ in a gradient descent method are given, and define $r_{k-1} = f - Ax_{k-1}$. Then, the step size can be computed exactly via the formula

$$\alpha_k = \arg\min_{\alpha\in\mathbb{C}} E_A(x_{k-1} + \alpha d_{k-1}) = \frac{d_{k-1}^H r_{k-1}}{d_{k-1}^H A d_{k-1}}.$$

Proof. Let $\alpha = s + it$, where $s, t \in \mathbb{R}$. Define the function

$$\begin{aligned}
g(s,t) &= E_A(x_{k-1} + \alpha d_{k-1}) \\
&= \frac{1}{2}(x_{k-1} + \alpha d_{k-1})^H A (x_{k-1} + \alpha d_{k-1}) - \Re((x_{k-1} + \alpha d_{k-1})^H f) \\
&= \frac{1}{2} x_{k-1}^H A x_{k-1} + \frac{\alpha}{2} x_{k-1}^H A d_{k-1} + \frac{\overline{\alpha}}{2} d_{k-1}^H A x_{k-1} + \frac{|\alpha|^2}{2} d_{k-1}^H A d_{k-1} - \Re((x_{k-1} + \alpha d_{k-1})^H f) \\
&= E_A(x_{k-1}) + \frac{|\alpha|^2}{2} d_{k-1}^H A d_{k-1} + \Re(\overline{\alpha} d_{k-1}^H A x_{k-1}) - \Re(\overline{\alpha} d_{k-1}^H f) \\
&= E_A(x_{k-1}) - \Re(\overline{\alpha} d_{k-1}^H r_{k-1}) + \frac{|\alpha|^2}{2} d_{k-1}^H A d_{k-1} \\
&= E_A(x_{k-1}) - s\Re(d_{k-1}^H r_{k-1}) - t\Im(d_{k-1}^H r_{k-1}) + \frac{s^2 + t^2}{2} d_{k-1}^H A d_{k-1}.
\end{aligned}$$

This is a strictly positive quadratic function of two variables. Setting the first derivatives equal to zero, we find

$$0 = \frac{\partial g}{\partial s}(s_k, t_k) = -\Re(d_{k-1}^H r_{k-1}) + s_k d_{k-1}^H A d_{k-1},$$

$$0 = \frac{\partial g}{\partial t}(s_k, t_k) = -\Im(d_{k-1}^H r_{k-1}) + t_k d_{k-1}^H A d_{k-1}$$

which implies that

$$\alpha_k = s_k + it_k = \frac{d_{k-1}^H r_{k-1}}{d_{k-1}^H A d_{k-1}}.$$

□

4.3 The steepest descent method

As we have seen in the previous section, the gradient descent method's search direction is supplied by the algorithm. For the steepest descent method we use as search direction the one given by the residual:

Definition 4.3. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, and $f \in \mathbb{C}^n$. The* ***steepest descent method*** *is a gradient descent method for which the search direction d_{k-1} is defined to be the residual,*

$$d_{k-1} = r_{k-1} = f - Ax_{k-1},$$

so that the step size is precisely

$$\alpha_k = \frac{r_{k-1}^H r_{k-1}}{r_{k-1}^H A r_{k-1}}.$$

If $L \in \mathbb{C}^{n\times n}$ is an HPD matrix, the ***L-preconditioned steepest descent*** *method is a gradient descent method with search direction*

$$d_{k-1} = L^{-1} r_{k-1},$$

so that the step size is precisely

$$\alpha_k = \frac{r_{k-1}^H L^{-1} r_{k-1}}{r_{k-1}^H L^{-1} A L^{-1} r_{k-1}}.$$

Proposition 4.3. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, and $f \in \mathbb{C}^n$. Suppose that $\{x_k\}_{k=1}^{\infty}$ is computed using the steepest descent method with the starting vector x_0. Then the sequence of residual vectors $\{r_k\}_{k=1}^{\infty}$, $r_k = f - Ax_k$, has the property that*

$$(r_k, r_{k+1})_2 = r_{k+1}^H r_k = 0,$$

for $k = 0, 1, 2, \ldots$.

Proof.

$$\begin{aligned}
(r_{k+1}, r_k)_2 = r_k^H r_{k+1} &= r_k^H (f - Ax_{k+1}) \\
&= r_k^H (f - A(x_k + \alpha_{k+1} d_k)) \\
&= r_k^H (f - Ax_k - \alpha_{k+1} A d_k) \\
&= r_k^H (r_k - \alpha_{k+1} A r_k) \\
&= r_k^H (r_k - \frac{r_k^H r_k}{r_k^H A r_k} A r_k) \\
&= r_k^H r_k - \frac{r_k^H r_k}{r_k^H A r_k} r_k^H A r_k \\
&= r_k^H r_k - r_k^H r_k \\
&= 0,
\end{aligned}$$

which completes the proof. □

As a consequence of the last proposition it follows that search directions are pairwise orthogonal. For the error in the steepest descent method we have:

Theorem 4.4. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n$, and $x = A^{-1}f$. Suppose that $\{x_k\}_{k=1}^{\infty}$ is computed using the steepest descent method with the starting value $x_0 \in \mathbb{C}^n$. Then the error $e_k = x - x_k$ satisfies*

$$\|e_{k+1}\|_A^2 = \gamma_k \|e_k\|_A^2,$$

where

$$\gamma_k = 1 - \frac{(r_k^H r_k)^2}{(r_k^H r_k)(r_k^H A^{-1} r_k)}.$$

Proof. Suppose $x = A^{-1}f$. Then, x minimizes the quadratic functional $E_A : \mathbb{C}^n \to \mathbb{R}$ defined by

$$E_A(z) = \frac{1}{2} z^H A z - \Re(z^H f),$$

so that the following holds for any $z \in \mathbb{C}^n$

$$E_A(z) = E_A(x) + \frac{1}{2}\|z - x\|_A^2. \tag{4.6}$$

We have that $r_k = f - Ax_k$, and the step size is

$$\alpha_{k+1} = \frac{r_k^H r_k}{r_k^H A r_k}.$$

Note that $\alpha_{k+1} = \overline{\alpha_{k+1}}$ and $r_k^H r_k = \overline{(r_k^H r_k)}$. We then conclude that

$$\begin{aligned}
E_A(x_{k+1}) &= E_A(x_k + \alpha_{k+1} r_k) = \frac{1}{2}(x_k^H + \alpha_{k+1} r_k^H) A (x_k + \alpha_{k+1} r_k) - \Re(x_k^H f + \alpha_{k+1} r_k^H f) \\
&= \frac{1}{2} x_k^H A x_k + \frac{1}{2}\alpha_{k+1} x_k^H A r_k + \frac{1}{2}\alpha_{k+1} r_k^H A x_k + \frac{1}{2}\alpha_{k+1}^2 r_k^H A r_k - \Re(x_k^H f) - \Re(\alpha_{k+1} r_k^H f) \\
&= E_A(x_k) + \frac{1}{2}\alpha_{k+1}(x_k^H A r_k + r_k^H A x_k) + \frac{1}{2}\alpha_{k+1}^2 r_k^H A r_k - \alpha_{k+1}\Re(r_k^H r_k) - \alpha_{k+1}\Re(r_k^H A x_k) \\
&= E_A(x_k) + \alpha_{k+1}\Re(r_k^H A x_k) + \frac{1}{2}\alpha_{k+1}^2 r_k^H A r_k - \alpha_{k+1} r_k^H r_k - \alpha_{k+1}\Re(r_k^H A x_k) \\
&= E_A(x_k) + \frac{1}{2}\alpha_{k+1}^2 r_k^H A r_k - \alpha_{k+1} r_k^H r_k \\
&= E_A(x_k) + \frac{1}{2}\alpha_{k+1}\frac{r_k^H r_k}{r_k^H A r_k} r_k^H A r_k - \alpha_{k+1} r_k^H r_k \\
&= E_A(x_k) - \frac{1}{2}\frac{(r_k^H r_k)^2}{r_k^H A r_k}.
\end{aligned} \tag{4.7}$$

Combining equations (4.6) and (4.7) we get

$$\begin{aligned}
E_A(x_{k+1}) = E_A(x) + \frac{1}{2}\|x_{k+1} - x\|_A^2 \quad &\Rightarrow \quad \|e_{k+1}\|_A^2 = 2(E_A(x_{k+1}) - E_A(x)) \\
&\Rightarrow \quad \|e_{k+1}\|_A^2 = 2(E_A(x_k) - \frac{1}{2}\frac{(r_k^H r_k)^2}{r_k^H A r_k} - E_A(x)) \\
&\Rightarrow \quad \|e_{k+1}\|_A^2 = 2(E_A(x_k) - E_A(x)) - \frac{(r_k^H r_k)^2}{r_k^H A r_k} \\
&\Rightarrow \quad \|e_{k+1}\|_A^2 = \|e_k\|_A^2 - \frac{(r_k^H r_k)^2}{r_k^H A r_k}.
\end{aligned} \tag{4.8}$$

Since $r_k = Ae_k$, we have

$$\begin{aligned}
\|e_k\|_A^2 &= e_k^H A e_k \\
&= e_k^H r_k \\
&= r_k^H A^{-1} r_k.
\end{aligned} \tag{4.9}$$

Thus, (4.8) becomes

$$\begin{aligned}\|e_{k+1}\|_A^2 &= \|e_k\|_A^2\left(1-\frac{(r_k^H r_k)^2}{(r_k^H A r_k)\|e_k\|_A^2}\right)\\ &= \|e_k\|_A^2\left(1-\frac{(r_k^H r_k)^2}{(r_k^H A r_k)(r_k^H A^{-1} r_k)}\right)\\ &= \gamma_k\|e_k\|_A^2,\end{aligned}$$

which finishes the proof. □

Lemma 2. *(Kantorovich Inequality). Let the matrix $A \in \mathbb{C}^{n\times n}$ be HPD with spectrum $\sigma(A) = \{\lambda_i\}_{i=1}^n$, with $0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$, and spectral condition number*

$$\kappa = \kappa_2(A) = \frac{\lambda_n}{\lambda_1}.$$

Then, for any $x \in \mathbb{C}_\star^n$,

$$\frac{(x^H A x)(x^H A^{-1} x)}{(x^H x)^2} \le \frac{1}{4}(\sqrt{\kappa}+\sqrt{\kappa^{-1}})^2.$$

Proof. Define $\mu = \sqrt{\lambda_1\lambda_n}$. Then

$$\kappa^{-\frac{1}{2}} = \frac{\sqrt{\lambda_1}}{\sqrt{\lambda_n}} = \frac{\lambda_1}{\sqrt{\lambda_1\lambda_n}} \le \frac{\lambda_i}{\mu} \le \frac{\lambda_n}{\mu} = \kappa^{\frac{1}{2}} \tag{4.10}$$

and

$$\kappa^{-\frac{1}{2}} = \frac{\sqrt{\lambda_1}}{\sqrt{\lambda_n}} = \frac{\sqrt{\lambda_1\lambda_n}}{\lambda_n} \le \frac{\mu}{\lambda_i} \le \frac{\mu}{\lambda_1} = \kappa^{\frac{1}{2}}.$$

Therefore, for all $i = 1,\cdots,n$,

$$2\kappa^{-\frac{1}{2}} \le \frac{\lambda_i}{\mu} + \frac{\mu}{\lambda_i} \le 2\kappa^{\frac{1}{2}}.$$

Now observe that the function

$$f(x) = x + \frac{1}{x}$$

is strictly decreasing on $(0,1)$ and strictly increasing on $(1,\infty)$. Set $x = \frac{\lambda_i}{\mu}$. Using (4.10), if

$$1 \le x = \frac{\lambda_i}{\mu} \le \kappa^{\frac{1}{2}},$$

then

$$2 = f(1) \le f\left(\frac{\lambda_i}{\mu}\right) \le f(\kappa^{\frac{1}{2}}).$$

This implies that

$$2 \le \frac{\lambda_i}{\mu} + \frac{\mu}{\lambda_i} \le \kappa^{\frac{1}{2}} + \kappa^{-\frac{1}{2}}.$$

On the other hand, if

$$\kappa^{-\frac{1}{2}} \le x = \frac{\lambda_1}{\mu} \le 1,$$

then

$$f(\kappa^{-\frac{1}{2}}) \ge f\left(\frac{\lambda_i}{\mu}\right) \ge f(1) = 2,$$

which implies that

$$\kappa^{-\frac{1}{2}} + \kappa^{\frac{1}{2}} \geq \frac{\lambda_i}{\mu} + \frac{\mu}{\lambda_i} \geq 2.$$

Therefore, it's true for every case that

$$2 \leq \frac{\lambda_i}{\mu} + \frac{\mu}{\lambda_i} \leq \kappa^{\frac{1}{2}} + \kappa^{-\frac{1}{2}}. \tag{4.11}$$

Suppose that (λ_i, w_i) is an eigenpair of A, where $\{w_i\}_{i=1}^n$ is an orthogonal basis for $\mathbb{C}^n$. Then

$$(\mu^{-1}A + \mu A^{-1})w_i = \left(\frac{\lambda_i}{\mu} + \frac{\mu}{\lambda_i}\right) w_i.$$

Let $x \in \mathbb{C}^n_\star$ be arbitrary. There exist unique constants $x_i \in \mathbb{C}$ such that

$$x = \sum_{i=1}^n c_i w_i.$$

Then

$$\begin{aligned}
\frac{1}{\mu} x^H A x + \mu x^H A^{-1} x &= \sum_{i=1}^n \sum_{j=1}^n \frac{1}{\mu} c_i \overline{c_j} w_i^H A w_j + \mu c_i \overline{c_j} w_i^H A^{-1} w_j \\
&= \sum_{i=1}^n |c_i|^2 \left(\frac{\lambda_i}{\mu} w_i^H w_i + \frac{\mu}{\lambda_i} w_i^H w_i\right) \\
&= \sum_{i=1}^n |c_i|^2 \left(\frac{\lambda_i}{\mu} + \frac{\mu}{\lambda_i}\right) \\
&\leq (\kappa^{\frac{1}{2}} + \kappa^{-\frac{1}{2}}) \|x\|_2^2.
\end{aligned}$$

From the inequality

$$|ab| \leq \frac{1}{2}|a|^2 + \frac{1}{2}|b|^2,$$

for any $a, b \in \mathbb{R}$, it follows that

$$ab \leq |ab| \leq \frac{1}{4}(|a| + |b|)^2.$$

Using the last inequality with

$$a = \frac{1}{\mu} x^H A x, \qquad b = \mu x^H A^{-1} x,$$

we obtain

$$(x^H A x)(x^H A^{-1} x) \leq \frac{1}{4}\left(\frac{1}{\mu} x^H A x + \mu x^H A^{-1} x\right)^2 \leq \frac{1}{4}\left(\kappa^{\frac{1}{2}} + \kappa^{-\frac{1}{2}}\right)^2 (x^H x)^2,$$

from which the assertion of the lemma follows readily. □

Theorem 4.5. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n$, and $x = A^{-1} f$. Suppose that $\{x_k\}_{k-1}^\infty$ is computed using the steepest descent method with the starting value $x_0 \in \mathbb{C}^n$. Then the following estimate holds*

$$\|e_k\|_A \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|e_0\|_A,$$

where $\kappa = \kappa_2(A)$.

Proof. From Theorem 3.4 we have

$$\begin{aligned}\|e_k\|_A^2 &= \gamma_{k-1}\|e_{k-1}\|_A^2\\ &= \gamma_{k-1}\gamma_{k-2}\|e_{k-2}\|_A^2\\ &= \cdots\\ &= \gamma_{k-1}\gamma_{k-2}\cdots\gamma_0\|e_0\|_A^2.\end{aligned}$$

Using the Kantorovich inequality (2) for γ_k we observe that

$$\begin{aligned}\gamma_k = 1-\frac{(r_k^H r_k)^2}{(r_k^H r_k)(r_k^H A^{-1} r_k)} \le 1-\frac{4}{(\kappa^{-\frac{1}{2}}+\kappa^{\frac{1}{2}})^2} &= \frac{\left(\frac{1}{\sqrt{\kappa}}+\sqrt{\kappa}\right)^2-4}{\left(\frac{1}{\sqrt{\kappa}}+\sqrt{\kappa}\right)^2}\\ &= \frac{\frac{1}{\kappa}+2+\kappa-4}{\left(\frac{1}{\sqrt{\kappa}}+\sqrt{\kappa}\right)^2}\\ &= \frac{\left(\sqrt{\kappa}-\frac{1}{\sqrt{\kappa}}\right)^2}{\left(\sqrt{\kappa}+\frac{1}{\sqrt{\kappa}}\right)^2}\\ &= \left(\frac{\kappa-1}{\kappa+1}\right)^2.\end{aligned}$$

So,

$$\|e_k\|_A^2 \le \left(\frac{\kappa-1}{\kappa+1}\right)^{2k}\|e_0\|_A^2 \implies \|e_k\|_A \le \left(\frac{\kappa-1}{\kappa+1}\right)^{k}\|e_0\|_A.$$

□

Remark 4.1. *We observe that, for large spectral condition number* κ

$$\frac{\kappa-1}{\kappa+1} \approx 1-\frac{2}{\kappa}.$$

In other words, the convergence rate deteriorates as $\kappa \to \infty$.

4.4 The conjugate gradient method

Recall that finding the solution of $Ax = f$, where $A \in \mathbb{C}^{n\times n}$ is Hermitian Positive Definite (HDP) and $f \in \mathbb{C}^n$ is equivalent to minimizing the functional

$$E_A(z) = \frac{1}{2}z^H A z - \Re(z^H f),$$

that is, finding

$$x = \arg\min_{z\in\mathbb{C}} E_A(z).$$

A smarter way to solve this system is to minimize the functional over a sequence of increasing larger subspaces of $\mathbb{C}^n$. We begin with some necessary definitions.

Definition 4.4. *Given* $A \in \mathbb{C}^{n\times n}$ *and* $\mathbf{0} \neq \mathbf{q} \in \mathbb{C}^n$, *the* ***Krylov subspace*** *of degree* m *is*

$$\mathscr{K}_m(A,\mathbf{q}) = span\{A^k\mathbf{q} | k = 0,\cdots,m-1\}.$$

Definition 4.5. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n_\star$, and $x = A^{-1}f$. The **zero-start conjugate gradient method** is an iterative scheme for producing a sequence of approximations $\{x_k\}_{k=1}^{\infty}$ from the starting point $\mathbf{0} = x_0 \in \mathbb{C}^n$ according to the following formula: setting $\mathscr{K}_k = \mathscr{K}_k(A,f)$, the $k-$th iterate is obtained by*

$$x_k = \underset{z\in\mathscr{K}_k}{\arg\min} E_A(z). \tag{4.12}$$

By construction, $\mathscr{K}_m(A,q) \subseteq \mathscr{K}_{m+1}(A,q)$. Thus, we are minimizing over a non-decreasing family of nested subspaces of $\mathbb{C}^n$.

Remark 4.2. *Someone may possibly wonder which is the most appropriate value for the starting vector x_0 such that the conjugate gradient method converges faster. Later we will see the importance of starting the method with a non-zero starting vector x_0.*

Definition 4.6. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n$, $x = A^{-1}f$, and W is a subspace of $\mathbb{C}^n$. The vector $x_W \in W$ is called the **Galerkin approximation** of x in W if and only if*

$$(Ax_W, w)_2 = (f,w)_2, \qquad \forall w \in W. \tag{4.13}$$

In the following theorem we prove the existence and uniqueness of the Galerkin approximation.

Theorem 4.6. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n$, $x = A^{-1}f$, and W is a subspace of $\mathbb{C}^n$. The Galerkin approximation $x_W \in W$ exists and is unique.*

Proof. Let $B = \{w_1, \cdots, w_k\}$ be an **A**-orthonormal basis for W, i.e.,

$$(w_i, w_j)_A = (Aw_i, w_j)_2 = \delta_{i,j},$$

for all $1 \le i, j \le k \le n$. Then (4.13) holds if and only if

$$(Ax_W, w_i)_2 = (f, w_i)_2, \qquad i = 1, \cdots, k. \tag{4.14}$$

Since B is a basis, there are unique constants $c_1, \cdots, c_k \in \mathbb{C}$ such that $x_W = \sum_{j=1}^{k} c_j w_j$. Plugging this into (4.14) we get

$$\begin{aligned} & (\sum_{j=1}^{k} c_j A w_j, w_i)_2 = (f, w_i)_2 \\ \Rightarrow \quad & \sum_{j=1}^{k} c_j (w_j, w_i)_A = (f, w_i)_2 \\ \Rightarrow \quad & c_i = (f, w_i)_2, \qquad i = 1, \ldots, k. \end{aligned}$$

Thus $c_i, \quad i = 1, \cdots, k$ exist and they are unique by definition. □

The next property summarizes some important properties of the Galerkin approximations.

Proposition 4.4. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n$, $x = A^{-1}f$, W is a subspace of $\mathbb{C}^n$, and x_W is the Galerkin approximation to x in W.*

1. *The residual is orthogonal to W. That is, if $r = f - Ax_W$, we have*

$$(r, w)_2 = 0, \qquad \forall w \in W.$$

2. *Galerkin orthogonality: Define the error $e = x - x_W$. Then we have*

$$(Ae, w)_2 = 0, \qquad \forall w \in W.$$

3. *Optimality:*
$$(Ae,e)_2 \leq (A(x-w),x-w)_2, \qquad \forall w \in W.$$

Proof. Since x_W is the Galerkin approximation to x in W, it's true that

$$(Ax_W,w)_2 = (f,w)_2, \quad \forall x \in W.$$

1. According to the last observation we have that

$$\begin{aligned}(r,w)_2 &= (f-Ax_W,w)_2\\ &= (f,w)_2-(Ax_W,w)_2\\ &= (f,w)_2-(f,w)_2\\ &= 0.\end{aligned}$$

2. For the proof of the Galerkin orthogonality, we have that

$$\begin{aligned}(Ae,w)_2 &= (A(x-x_W),w)_2\\ &= (Ax,w)_2-(Ax_W,w)_2\\ &= (Ax,w)_2-(f,w)_2\\ &= (Ax-f,w)_2\\ &= 0.\end{aligned}$$

3. Before proving the optimality of the Galerkin orthogonality, we need to recall some important properties. Firstly, we know that if W is a subspace of $\mathbb{C}^n$ then for every $x,y \in W$ it's also true that $x-y \in W$. Secondly, the Cauchy-Schwarz inequality for the A-norm says that for any $u,v \in \mathbb{C}^n$

$$|(u,v)_A| \leq \|u\|_A\|v\|_A. \tag{4.15}$$

So we have that
$$(Ae,x-y)_2 = 0, \qquad \forall x,y \in W.$$

Therefore,

$$\begin{aligned}\|e\|_A^2 = (Ae,e)_2 &= (Ae,x-x_W)_2\\ &= (Ae,e)_2\\ &= (Ae,x-x_W)_2+(Ae,x_W-w)_2\\ &= (Ae,x-x_W+x_W-w)_2\\ &= (Ae,x-w)_2\\ &= (e,x-w)_A \leq \|e\|_A\|x-w\|_A.\end{aligned}$$

In the case of $\|e\|_A = 0$, the result is trivial, while in the case of $\|e\|_A > 0$, we have

$$\|e\|_A \leq \|x-w\|_A,$$

and the result follows.

□

Theorem 4.7. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n$, and W is a subspace of $\mathbb{C}^n$. Then, the following are equivalent:*

1. *The vector $x_W \in W$ is a minimizer of E_A over W:*

$$x_W = \arg\min_{z \in W} E_A(z).$$

2. *The vector $x_W \in W$ is a Galerkin approximation of $x = A^{-1}f$:*

$$(Ax_W, w)_2 = (f, w)_2, \qquad \forall w \in W.$$

Theorem 4.8. *Let A be HPD, $f \in \mathbb{C}^n_\star$, and $x = A^{-1}f$. We have that $\dim \mathscr{K}_m(A, f) = m$ and, as a consequence, the sequence $\{x_k\}_{k=1}^\infty$, generated by the zero-start conjugate gradient method is such that there is an integer $m_\star \in \{1, \cdots, n\}$ for which*

$$x_k \neq x, \quad k = 1, \ldots, m_\star - 1, \qquad x_k = x, \quad k \geq m_\star.$$

Proof. Let $\mathscr{K}_m = \mathscr{K}_m(A, f)$. Notice that $\dim \mathscr{K}_m \leq m$. We have to show that the equality $dim \mathscr{K}_m = m$ holds. We will proceed by induction. Set $m = 1, \cdots, k$ and notice that, since $f \neq 0$,

$$\mathscr{K}_1 = span\{f\} \Rightarrow \dim \mathscr{K}_1 = 1.$$

Assume that for all $m = 1, \ldots, k$ with $k < n - 1$ we have $\dim \mathscr{K}_k = k$ and $x_k \neq x$. Therefore, the residual $r_k = f - Ax_k \neq 0$. Observe that $x_k = \arg\min_{z \in \mathscr{K}_k} E_A(z)$ and so $x_k \in \mathscr{K}_k$. So, $Ax_k \in \mathscr{K}_{k+1}$ and the corollary to this is that $r_k \in \mathscr{K}_{k+1}$. Now we have that $x_k \in \mathscr{K}_k$ is a minimizer of E_A over $\mathscr{K}_k$, so using Theorem 4.7 we observe that

$$(r_k, w)_2 = (f - Ax_k, w)_2 = (f, w)_2 - (Ax_k, w)_2 = 0 \qquad \forall x_k \in \mathscr{K}_k.$$

Thus the residual r_k must be orthogonal to $\mathscr{K}_k$. The last equality above shows that $r_k \neq 0$ and by extension $r_k \in \mathscr{K}_{k+1} \backslash \mathscr{K}_k$. The last conclusion is only possible if $\dim \mathscr{K}_{k+1} > \dim \mathscr{K}_k$. As a consequence $\dim \mathscr{K}_{k+1} = k + 1$. □

Theorem 4.8 shows that the exact solution can be found in at most n steps. This can be expressed as an iterative scheme for two reasons. First, we should avoid the case of a very large n and so the method could stop after a specific number of iterations. Second, experience shows that rounding errors make the calculation of the exact solution almost impossible.

We going to rephrase the conjugate gradient method in a more convenient equivalent form. As stated in Definition 4.12, at step k we need to store k vectors of a basis of $\mathscr{K}_k(A, f)$, as we need to minimize over it. The theorem below shows that this action is not necessary.

Theorem 4.9. *Suppose that $A \in \mathbb{C}^{n \times n}$ is HPD, $f \in \mathbb{C}^n_\star$, and $x = A^{-1}f$. The sequence generated by the zero-start conjugate gradient method, $\{x_k\}_{k=1}^{m_\star}$, is the same sequence as that generated by the following recursive algorithm:*

```
x_0 = 0, r_0 = f - Ax_0 = f, p_0 = r_0 = f
for k = 0, 1, ···, m_⋆ - 1 do
    if k ≥ 1 then
        p_k = r_k - μ_k p_{k-1}
        μ_k = (Ar_k, p_{k-1})_2 / (Ap_{k-1}, p_{k-1})_2
    end
    x_{k+1} = x_k + λ_{k+1} p_k
    λ_{k+1} = (r_k, p_k)_2 / (Ap_k, p_k)_2
    r_{k+1} = r_k - λ_{k+1} A p_k
end
if k = m_⋆ - 1 then
    p_{m_⋆} = 0
end
```

Clearly,

$$r_{m_\star} = 0 = p_{m_\star}.$$

In addition,

$$0 \neq r_k \in \mathscr{K}_{k+1} \backslash \mathscr{K}_k, \quad 0 \neq p_k \in \mathscr{K}_{k+1} \backslash \mathscr{K}_k, \qquad k = 0, \cdots, m_\star - 1,$$

and

$$(r_j, r_i)_2 = (p_j, p_i)_A = 0,$$

for all $0 \leq j < i \leq m_\star - 1$.

Proof. Suppose that the solution is obtained after $m_\star - 1$ iterations. We will prove the theorem by induction on k. The cases for $k \leq 2$ are clearly true. So we have the induction hypothesis that the formulae and properties of the theorem are true for $0 \leq k \leq m-1$. Therefore, we are going to study the case of $k \leq m \leq m_\star - 1$.

Assume that $\{x_k\}_{k=1}^{m+1}$ is generated by the zero-start conjugate gradient algorithm. It's true that $x_m \neq x$ and $r_m = f - Ax_m \neq 0$. In addition, $x_{k+1} \in \mathscr{K}_{k+1}$ where x_{k+1} is given by (4.12). As we concluded in the proof of Theorem 4.8, $r_m \in \mathscr{K}_m$. Since x_m is a Galerkin approximation to x, the Galerkin orthogonality (4.7) shows that

$$\begin{aligned} 0 = (Ae_m, y)_2 &= (A(x - x_m), y)_2 && (4.16)\\ &= (Ax, y)_2 - (Ax_m, y)_2 && (4.17)\\ &= (f, y)_2 - (Ax_m, y)_2 && (4.18)\\ &= (f - Ax_m, y)_2 && (4.19)\\ &= (r_m, y)_2, && (4.20) \end{aligned}$$

where $y \in \mathscr{K}_m$. So r_m is orthogonal to an arbitrary component of $\mathscr{K}_m$, that is $r_m \in \mathscr{K}_{m+1} \backslash \mathscr{K}_m$. The induction hypothesis guarantees that $\{r_0, \cdots, r_{m-1}\}$ is an orthogonal set and a basis for $\mathscr{K}_m$. Also, the Galerkin orthogonality implies that r_m is orthogonal to this basis so $\{r_0, \cdots, r_{m-1}, r_m\}$ is an orthogonal basis for $\mathscr{K}_{m+1}$.

Denote by $\mathscr{K}_m^\perp$ the orthogonal complement of $\mathscr{K}_m$ in $\mathscr{K}_{m+1}$ in the A-inner product, that is,

$$\mathscr{K}_m^\perp = \{w \in \mathscr{K}_{m+1} \,|\, (Aw, y)_2 = 0, \forall y \in \mathscr{K}_m\}.$$

Observe that $r_m \in \mathscr{K}_m^\perp$ so $\mathscr{K}_m^\perp \neq \{\mathbf{0}\}$. Notice that $\mathscr{K}_{m+1}$ can be written as the direct sum of its subspaces $\mathscr{K}_m$ and $\mathscr{K}_m^\perp$, i.e.

$$\mathscr{K}_{m+1} = \mathscr{K}_m \overset{\perp}{\oplus} \mathscr{K}_m^\perp.$$

From the properties of the direct sum, we have that

$$\dim(\mathscr{K}_{m+1}) = \dim(\mathscr{K}_m) + \dim(\mathscr{K}_m^\perp).$$

Since $\dim(\mathscr{K}_{m+1}) = m+1$ and $\dim(\mathscr{K}_m) = m$ we conclude that $\dim(\mathscr{K}_m^\perp) = 1$. Thus any element $\xi_{m+1} \in \mathscr{K}_{m+1}$ can be written as

$$\xi_{m+1} = \xi_m + \mu p_m,$$

for some $\mu \in \mathbb{C}$, $p_m \in \mathscr{K}_m^\perp \cap \mathbb{C}_\star^n$, and $\xi_m \in \mathscr{K}_m$.

Now consider the element

$$w = x_m + \lambda_{m+1} p_m, \qquad \lambda_{m+1} = \frac{(r_m, p_m)_2}{(Ap_m, p_m)_2}.$$

We observe the following:

1. $r_m \in \mathscr{K}_{m+1} \backslash \mathscr{K}_m, \quad \xi_m \in \mathscr{K}_m$ so $r_m \perp \xi_m$ and $(r_m, \xi_m)_2 = 0$.
2. $p_m \in \mathscr{K}_m^{\perp}$ so from the definition of $\mathscr{K}_m^{\perp}$ we have that $(Ap_m, \xi_m)_2 = 0$ since $\xi_m \in \mathscr{K}_m$.

Then,

$$\begin{aligned}
(Aw - f, \xi_{m+1})_2 &= (A(x_m + \lambda_{m+1} p_m) - f, \xi_m + \mu p_m)_2 \\
&= (Ax_m - f, \xi_m)_2 + \overline{\mu}(Ax_m - f, p_m)_2 + \lambda_{m+1}(Ap_m, \xi_m)_2 + \overline{\mu}\lambda_{m+1}(Ap_m, p_m)_2 \\
&= -(r_m, \xi_m)_2 - \overline{\mu}[(r_m, p_m)_2 - \lambda_{m+1}(Ap_m, p_m)_2] \\
&= -\overline{\mu}[(r_m, p_m)_2 - \lambda_{m+1}(Ap_m, p_m)_2] \\
&= -\overline{\mu}\,[(r_m, p_m)_2 - \frac{(r_m, p_m)_2}{(Ap_m, p_m)_2}(Ap_m, p_m)_2] \\
&= 0.
\end{aligned}$$

Therefore,

$$(Aw - f, \xi_{m+1})_2 = 0, \qquad \forall \xi_{m+1} \in \mathscr{K}_{m+1}.$$

But $x_{m+1} \in \mathscr{K}_{m+1}$ is the unique element that has this property in the last equation. Therefore, $w = x_{m+1}$. In other words, $x_{m+1} \in \mathscr{K}_{m+1}$ is the Galerkin approximation defining the $(m+1)$-st iterate in the zero-start conjugate gradient algorithm if and only if

$$x_{m+1} = x_m + \lambda_{m+1} p_m, \qquad \lambda_{m+1} = \frac{(r_m, p_m)_2}{(Ap_m, p_m)_2}.$$

From the last equation we can compute the $(m+1)$-st residual:

$$\begin{aligned}
r_{m+1} &= f - Ax_{m+1} \\
&= f - Ax_m - \lambda_{m+1} Ap_m \\
&= r_m - \lambda_{m+1} Ap_m
\end{aligned}$$

By the induction hypothesis, $\{p_0, \cdots, p_{m-1}\}$ is an A-orthogonal set and, hence, it forms a basis for $\mathscr{K}_m$. Since $r_m \in \mathscr{K}_{m+1}$ has a component in $\mathscr{K}_m^{\perp}$, its A-orthogonal projection, q, into $\mathscr{K}_m^{\perp}$ is non-zero:

$$q = r_m - \sum_{i=0}^{m-1} \frac{(Ar_m, p_i)_2}{(Ap_i, p_i)_2} p_i = r_m - \sum_{i=0}^{m-1} \frac{(r_m, p_i)_A}{(p_i, p_i)_A} p_i. \tag{4.21}$$

But for $0 \le i \le m-2$ we have $Ap_i \in \mathscr{K}_m$ so $(Ap_i, r_m)_2 = 0$. Thus, equation (4.21) becomes

$$q = r_m - \frac{(Ar_m, p_{m-1})_2}{(Ap_{m-1}, p_{m-1})_2} p_{m-1} \in \mathscr{K}_m^{\perp}.$$

Taking

$$p_m = q = r_m - \frac{(Ar_m, p_{m-1})_2}{(Ap_{m-1}, p_{m-1})_2} p_{m-1} \in \mathscr{K}_m^{\perp},$$

it follows that $\{p_0, \cdots, p_{m-1}, p_m\}$ is an A-orthogonal set. □

Considering all of the above, we may consider asking how to choose between a direct method (Cholesky factorization) and an iterative method (Conjugate Gradient) when we have a large sparse matrix? Some answers arise from the following example.

Example 4.4.1. *Suppose the system $Ax = b$, where A is a symmetric positive definite matrix. First we are going to solve it using the Cholesky Factorization method and then using the Conjugate Gradient method. We use the matrices "bcsstk01" and "bcsstk16". The next table sums up the results.*

Table 4.1: Results for bcsstk01

	Choleksy Factorization	Conjugate Gradient method
Convergence	True	True
Time	0.0003273	0.0041463
Residual error	5.658646e-13	6.573808e-05

Table 4.2: Results for bcsstk16

	Choleksy Factorization	Conjugate Gradient method
Convergence	True	True
Time	0.1650811	0.1376722
Residual error	3.774699e-12	0.000628

Observe that the Cholesky Factorization converges to a solution with a residual error less than 10^{-4} *while the Conjugate Gradient method has a bigger residual error.*

Corollary 4.1. *Suppose that* $A \in \mathbb{C}^{n\times n}$ *is HPD,* $f \in \mathbb{C}^n_\star$ *and* $x = A^{-1}f$*. The sequence generated by the zero-start conjugate gradient method,* $\{x_k\}_{k=1}^{m_\star}$ *has the following property for all* $k \in \{1,\cdots,m_\star\}$*,*

$$x_k \in \mathscr{K}_k \backslash \mathscr{K}_{k-1}.$$

This implies that

$$\langle x_1,\cdots,x_k\rangle = \mathscr{K}_k.$$

Corollary 4.2. *Suppose that* $A \in \mathbb{C}^{n\times n}$ *is HPD,* $f \in \mathbb{C}^n_\star$ *and* $x = A^{-1}f$*. If the zero-start conjugate gradient (CG) algorithm is employed to produce the approximation sequence* $\{x_j\}_{j=1}^{m_\star}$*, then, for all* $1 \le i \le m_\star$

$$\begin{aligned}
\mathscr{K}_i(A,f) &= \langle f, Af, \cdots, A^{i-1}f\rangle && (4.22)\\
&= \langle x_1,\cdots,x_i\rangle && (4.23)\\
&= \langle p_0,\cdots,p_{i-1}\rangle && (4.24)\\
&= \langle r_0,\cdots,r_{i-1}\rangle && (4.25)
\end{aligned}$$

Theorem 4.10. *For* $A \in \mathbb{C}^{n\times n}$ *HPD,* $f \in \mathbb{C}^n_\star$ *given and* $x = A^{-1}f$*, let* $\{x_i\}_{i=0}^m$ *for* $m \in \{1,\cdots,n\}$ *denote any sequence of vectors with* $x_0 = 0$ *that satisfies:*

$$\mathscr{K}_j = \mathscr{K}_j(f,A) = \langle f, Af, \cdots, A^{j-1}f\rangle = \langle x_1,\cdots,x_j\rangle = \langle r_0,\cdots,r_{j-1}\rangle, \quad r_{j-1} \neq 0$$

for all $j = 1,\ldots,n$*, with orthogonality relations:*

$$r_k^H r_l = 0 \quad \forall\ 0 \le k < l \le m. \tag{4.26}$$

Then the jth *iterate* x_j *is the unique vector in* $\mathscr{K}_j$ *that minimizes the error function* $\phi(y) = \|x-y\|_A$*. Furthermore,* ϕ *is monotonically decreasing:*

$$\|e_j\|_A = \|x-x_j\|_A = \phi(x_j) \le \phi(x_{j-1}) = \|x-x_{j-1}\|_A = \|e_{j-1}\|_A$$

Proof. Let $z \in \mathscr{K}_j$ be arbitrary. Define $w = x_j - z \in \mathscr{K}_i$, then:

$$\begin{aligned}
\phi^2(z) &= \|x-z\|_A^2 = \|x-x_j+x_j-z\|_A^2 = \|e_j+w\|_A^2 = (e_j+w)^H A(e_j+w) \\
&= \|e_j\|_A^2 + w^H A e_j + e_j^H A w + \|w\|_A^2 \\
&= \|e_j\|_A^2 + w^H r_j + (Ae_j)^H w + \|w\|_A^2 \\
&= \|e_j\|_A^2 + w^H r_j + r_j^H w + \|w\|_A^2 \\
&= \|e_j\|_A^2 + 2\Re(w^H r_j) + \|w\|_A^2,
\end{aligned}$$

where $r_j = f - Ax_j = Ax - Ax_j = Ae_j$. Since $w \in \mathscr{K}_j = \langle r_0, \cdots, r_{j-1}\rangle$, there exists unique $\alpha_0, \cdots, \alpha_{j-1} \in \mathbb{C}$ such that

$$w = \sum_{i=0}^{j-1} \alpha_i r_i.$$

So

$$w^H r_j = \sum_{i=0}^{j-1} \overline{\alpha_i} r_i^H r_j$$

and the fact that $r_k^H r_l = 0 \quad \forall \quad 0 \le k < l \le m$ follows from $w^H r_j = 0$. Hence

$$\phi^2(z) = \|e_j\|_A^2 + \|w\|_A^2 \ge \|e_j\|_A^2.$$

The equality holds for $w = 0$, so $z = x_j$. Hence x_j ia the unique minimizer of ϕ over $\mathscr{K}_j$.
Since $\mathscr{K}_{j-1}(f,A) \subseteq \mathscr{K}_j(f,A)$ we have the result

$$\|e_j\|_A = \phi(x_j) = \inf\{\phi(z) | z \in \mathscr{K}_j(f,A)\} \le \inf\{\phi(z) | z \in \mathscr{K}_{j-1}(f,A)\} = \phi(x_{j-1}) = \|e_{j-1}\|_A$$

□

Theorem 4.11. *Suppose that the zero-start conjugate gradient method is applied to solve $Ax = f$, where $A \in \mathbb{C}^{n\times n}$ is HPD and $f \in \mathbb{C}^n_\star$. Then, if the iteration has not already converged ($r_{i-1} \neq 0$), then there is a unique polynomial*

$$p_i \in \mathbb{P}_i^\star = \{p \in \mathbb{P}_i \,|\, p(0) = 1\}$$

that minimizes $\|p(A)e_0\|_A$ over all $p \in \mathbb{P}_i^\star$. The iterate x_i has the error $e_i = p_i(A)e_0$ and consequently,

$$\frac{\|e_i\|_A}{\|e_0\|_A} \le \inf_{p\in\mathbb{P}_i^\star} \max_{\lambda\in\sigma(A)} |p(\lambda)| \tag{4.27}$$

Proof. We have

$$E_A(z) = \frac{1}{2} z^H A z - \Re(z^H f) = \frac{1}{2}\|z\|_A^2 - \frac{1}{2}(z^H f + f^H z)$$

and

$$\begin{aligned}
\frac{1}{2}\|z-x\|_A^2 - \frac{1}{2} f^H A^{-1} f &= \frac{1}{2}(z-x)^H A(z-x) - \frac{1}{2} f^H A^{-1} f \\
&= \frac{1}{2} z^H A z - \frac{1}{2} z^H A x - \frac{1}{2} x^H A z + \frac{1}{2} x^H A x - \frac{1}{2} f^H A^{-1} f \\
&= \frac{1}{2} z^H A z - \Re(z^H f) + \frac{1}{2}\|x\|_A^2 - \frac{1}{2} f^H A^{-1} f \\
&= \frac{1}{2} z^H A z - \Re(z^H f) + \frac{1}{2}\|A^{-1} f\|_A^2 - \frac{1}{2} f^H A^{-1} f \\
&= \frac{1}{2}\|z\|_A^2 - \Re(z^H f) + \frac{1}{2} f^H A^{-1} A A^{-1} f - \frac{1}{2} f^H A^{-1} f \\
&= \frac{1}{2}\|z\|_A^2 - \Re(z^H f) \\
&= E_A(z).
\end{aligned}$$

Therefore, by the definition of the zero-start CG method, we have

$$\begin{aligned} x_i &= \underset{z\in\mathscr{K}_i}{argmin}\|x-z\|_A \qquad\qquad \underset{z\in\mathscr{K}_i}{min}\|x-z\|_A = \|x-x_i\|_A = \|e_i\|_A \\ &= \underset{z\in\mathscr{K}_i}{argmin}E_A(z) \\ &= \underset{z\in\mathscr{K}_i}{argmin}\frac{1}{2}\|z-x\|_A^2 - \frac{1}{2}f^H A^{-1} f. \end{aligned}$$

Since $x_0 = 0$ we have $e_0 = x$ and consequently, $r_0 = f - Ax_0 = f$. For any $z \in \mathscr{K}_i$, there are constants $c_j \in \mathbb{C}$, $1 \le j \le i$ such that:

$$z = \sum_{j=1}^{i}(-c_j)A^{j-1}f \Rightarrow x - z = x + \sum_{j=1}^{i}(-c_j)A^{j-1}f.$$

It follows that,

$$\begin{aligned} x - z &= e_0 + \sum_{j=1}^{i}(-c_j)A^{j-1}r_0 \\ &= e_0 + \sum_{j=1}^{i}(-c_j)A^{j}A^{-1}r_0 \\ &= e_0[I + \sum_{j=1}^{i}c_jA^j] \\ &= e_0 p(A), \end{aligned}$$

where

$$p(x) = 1 + \sum_{j=1}^{i}c_jx^j \quad \in \mathbb{P}_i^\star = \{p \in \mathbb{P}_i : p(0) = 1\}. \tag{4.28}$$

It follows that the minimization problem is equivalent to minimize the above polynomial, i.e. is equivalent to

$$p_i = \underset{p\in\mathbb{P}_i^\star}{argmin}\|p(A)e_0\|_A \qquad \underset{p\in\mathbb{P}_\sqsupset^\star}{min}\|p(A)e_0\|_A = \underset{z\in\mathbb{K}_i}{min}\|x-z\|_A = \|e_i\|_A$$

Therefore,

$$\|e_i\|_A = \inf_{p\in\mathbb{P}_i^\star}\|p(A)e_0\|_A \le \inf_{p\in\mathbb{P}_i^\star}\|p(A)\|_A\|e_0\|_A$$

which implies that

$$\frac{\|e_i\|_A}{\|e_0\|_A} \le \inf_{p\in\mathbb{P}_i^\star}\|p(A)\|_A. \tag{4.29}$$

Suppose that $z \in \mathbb{C}_\star^n$ and let $\{w_1, \ldots, w_n\}$ be an orthonormal eigenbasis of A. Let $\sigma(A) = \{\lambda_1, \cdots, \lambda_n\}$ denote the set of eigenvalues of A with

$$Aw_j = \lambda_j w_j, \qquad j = 1, \ldots, n$$

Thus, z can be expressed as

$$z = \sum_{j=1}^{n}\alpha_j w_j,$$

for some constants $\alpha_j \in \mathbb{C}$, $\quad j = 1, \cdots, n$. Observe that

$$Az = \sum_{j=1}^{n}\alpha_j A w_j = \sum_{j=1}^{n}\alpha_j\lambda_j w_j$$

and $(w_j, w_i)_2 = 0 \ \ \forall i \neq j$. Therefore

$$\|z\|_A^2 = z^H A z = \sum_{j=1}^{n} \overline{\alpha_j} w_j^H \alpha_j \lambda_j w_j = \sum_{j=1}^{n} |\alpha_j|^2 \lambda_j. \tag{4.30}$$

From (4.28) we claim that

$$\begin{aligned} p(A)w_j &= w_j + \sum_{k=1}^{i} c_k A^k w_j = w_j + \sum_{k=1}^{i} c_k \lambda_j^k w_j \\ p(\lambda_j)w_j &= w_j + \sum_{k=1}^{i} c_k \lambda_j^k w_j \end{aligned}$$

In the same way as in (4.30) we have

$$\begin{aligned} \|p(A)z\|_A^2 &= z^H p(A)^H A p(A) z \\ &= \sum_{j=1}^{n} \overline{\alpha_j} w_j^H p(A)^H A p(A) \alpha_j w_j \\ &= \sum_{j=1}^{n} |\alpha_j|^2 p(\lambda_j)^H w_j^H A p(\lambda_j) w_j \\ &= \sum_{j=1}^{n} |\alpha_j|^2 |p(\lambda_j)|^2 w_j^H A w_j \\ &= \sum_{j=1}^{n} |\alpha_j|^2 |p(\lambda_j)|^2 \lambda_j. \end{aligned}$$

The last equation divided by $\|z\|_A^2$ becomes,

$$\frac{\|p(A)z\|_A^2}{\|z\|_A^2} = \frac{\sum_{j=1}^{n} |\alpha_j|^2 |p(\lambda_j)|^2 \lambda_j}{\sum_{j=1}^{n} |\alpha_j|^2 \lambda_j} \leq \max_{\lambda \in \sigma(A)} |p(\lambda)|^2.$$

In conclusion

$$\|p(A)\|_A \leq \max_{\lambda \in \sigma(A)} |p(\lambda)| \tag{4.31}$$

Combining (4.31) and (4.29) we have the desired result

$$\frac{\|e_i\|_A}{\|e_0\|_A} \leq \inf_{p \in \mathbb{P}_i^\star} \|p(A)\|_A \leq \inf_{p \in \mathbb{P}_i^\star} \max_{\lambda \in \sigma(A)} |p(\lambda)|.$$

□

The next theorem is required to define the well-known *Chebyshev Polynomials*. For this reason, we are going to give a summary of Chebyshev Polynomials referring to the definition as well as some results about them.

Definition 4.7. *Let $x \in [-1, 1]$ and $x = \cos(\theta)$. The polynomial*

$$T_k(x) = \cos(k\theta), \quad k \in \mathbb{N}_0 \tag{4.32}$$

*is called the **Chebyshev Polynomial** of degree k. In the case of $|x| > 1$, the polynomial is*

$$T_k(x) = \cosh(k\theta), \quad k \in \mathbb{N}_0 \tag{4.33}$$

where $x = \cosh(\theta)$.

Proposition 4.5. *The Chebyshev polynomials satisfy the following three-term recurrence relation*

$$T_{n+1} = 2xT_n(x) - T_{n-1}(x), \qquad n = 1, 2, \ldots,$$

with $T_0(x) = 1$ *and* $T_1(x) = x \quad \forall x \in [-1, 1]$. *In addition* $T_k(1) = 1$ *and* $T_k(-1) = (-1)^k$.

Proof. Consider the trigonometric formula

$$\cos((n \pm 1)\theta) = \cos(n\theta)\cos(\theta) \mp \sin(n\theta)\sin(\theta).$$

Then

$$\begin{aligned} T_{n+1}(x) + T_{n-1}(x) &= \cos[(n+1)\theta] + \cos[(n-1)\theta] \\ &= \cos n\theta \cos\theta - \sin n\theta \sin\theta + \cos n\theta \cos\theta + \sin n\theta \sin\theta \\ &= 2\cos n\theta \cos\theta \\ &= 2\cos\theta T_n(x) \\ &= 2xT_n(x). \end{aligned}$$

Now, we have $x = \cos\theta \Rightarrow \theta = \cos^{-1}(x)$ so, for $x \in [-1, 1] \Rightarrow \theta \in [0, \pi]$ and the following relations hold

$$\begin{aligned} T_k(x) &= \cos(k\cos^{-1}(x)), \\ T_k(1) &= \cos(k\cos^{-1}(1)) = \cos(k2n\pi) = 1, \quad n \in \mathbb{Z}, \\ T_k(-1) &= \cos(k\cos^{-1}(-1)) = \cos(k(2n+1)\pi) = (-1)^k \in \mathbb{Z}. \end{aligned}$$

This completes the proof. □

Definition 4.8. *Let* $f \in \mathbb{C}^0([a,b], \mathbb{R})$ *be given and* $n \in \mathbb{N}_0 := \mathbb{N} \cup \{0\}$ *be fixed. The polynomial* $p \in \mathbb{P}_n$ *is the* ***best polynomial approximation*** *of* f *in* $L^\infty([a,b])$ *norm or* ***a minmax polynomial*** *if and only if*

$$\|f - p\|_\infty = \inf_{q \in \mathbb{P}_n} \|f - q\|_\infty \tag{4.34}$$

Definition 4.9. *Let* $f(x) \in C[a,b]$. *Then,*

1. $x \in [a,b]$ *is called a* $(+)$*-point for* $f(x)$ *if* $f(x) = \|f\|_\infty$
2. $x \in [a,b]$ *is called a* $(-)$*-point for* $f(x)$ *if* $f(x) = -\|f\|_\infty$
3. *A set of distinct points* $a \le x_0 < x_1 < \cdots < x_n \le b$ *is called an* ***alternating set*** *for* $f(x)$ *if the* x_i*'s are alternately* $(+)$*-points and* $(-)$*-points, that is, if* $|f(x_i)| = \|f\|_\infty$ *and* $f(x_i) = -f(x_{i-1})$ *for all* $i = 1, \ldots, n$.

Now we are able to state and prove the following theorem:

Theorem 4.12. *For any* $n \le 1$, *the formula* $p(x) = x^n - 2^{-n+1}T_n(x)$ *defines a polynomial* $p \in \mathbb{P}_{n-1}$ *satisfying*

$$2^{-n+1} = \max_{|x| \le 1} |x^n - p(x)| < \max_{|x| \le 1} |x^n - q(x)|, \tag{4.35}$$

for any other $q \in \mathbb{P}_{n-1}$, *i.e.* $p(x)$ *is the minmax polynomial for* $f(x) = x^n$.

Proof. The definition of minmax polynomial implies that

$$\|f - p\|_\infty = \inf_{q \in \mathbb{P}_n} \|f - q\|_\infty, \qquad q \in \mathbb{P}_n,$$

that is

$$\max_{|x| \le 1} |x^n - p(x)| = \|f - p\|_\infty = \|2^{-n+1}T_n(x)\|_\infty \le 2^{1-n}\|T_n(x)\|_\infty = 2^{1-n}.$$

Thus, if we show that $2^{1-n} < \underset{|x|\leq 1}{max}|x^n - q(x)|$ for any other $q \in \mathbb{P}_{n-1}$, then $p(x)$ is the minmax polynomial of f. Suppose p is the minmax polynomial of x^n. If q is another minmax polynomial of x^n then,

$$|x^n - q| > |x^n - p| \Rightarrow \max_{|x|\leq 1}|x^n - p| < \max_{|x|\leq 1}|x^n - q|$$

We have that the coefficient of x^n in the expression $2^{-n+1}T_n(x)$ is 1 because of the following:

$$\begin{aligned}
T_0(x) &= 1 \\
T_1(x) &= x \\
T_2(x) &= 2xT_1(x) - T_0(x) = 2x^2 - 1 \\
T_3(x) &= 2xT_2(x) - T_1(x) = 2^2x^3 - 3x \\
T_4(x) &= 2xT_3(x) - T_2(x) = 2^3x^4 - 8x^2 + 1 \\
&\vdots \\
T_n(x) &= 2xT_{n-1} - T_{n-2}(x) = 2^{n-1}x^n - \cdots
\end{aligned}$$

Therefore,

$$2^{1-n}T_n(x) = 2^{1-n}2^{n-1}x^n - \cdots$$

and

$$p(x) = x^n - 2^{1-n}T_n(x) \in \mathbb{P}_{n-1}.$$

Setting $x_k = \cos((n-k)\frac{\pi}{n})$ for $k = 0, 1, \ldots, n$. Then $-1 = x_0 < x_1 < \cdots < x_n = 1$ and

$$\begin{aligned}
T_n(x) &= T_n(\cos[(n-k)\frac{\pi}{n}]) \\
&= \cos(n(n-k)\frac{\pi}{n}) \\
&= \cos((n-k)\pi) \\
&= (-1)^{n-k}.
\end{aligned}$$

Also we have that

$$|T_n(x)| = |T_n(\cos\theta)| = |\cos(n\theta)| \leq 1, \quad \text{for } -1 \leq x \leq 1.$$

From the Definition 4.9, observe that $x_0 < \cdots < x_n$ is an alternating set containing $n+1$ points. Thus, using the next theorem, p must be the best approximation to x^n out of $\mathbb{P}_{n-2}$. □

Theorem 4.13. *Let $f \in C[a,b]$ and let $p \in \mathbb{P}_n$. If $f - p$ has an alternating set containing $n+2$ (or more) points, then p is the best approximation to f out of $\mathbb{P}_n$.*

Proof. Let $x_0, x_1, \ldots, x_{n+1}$ be an alternating set for $f - p$, and suppose that some $q \in \mathbb{P}_n$ is a better approximation to f than p, that is, $\|f - q\| < \|f - p\|$. In particular, we must have

$$|f(x_i) - p(x_i)| = \|f - p\| > \|f - q\| = |f(x_i) - q(x_i)|,$$

for $i = 0, \ldots, n+1$. The inequality $|f - p| > |f - q|$ implies that $q - p$ has the same sign as $f - p$. So,

$$q - p = (f - p) - (f - q)$$

alternates in sign $n+2$ times (because $f - p$ does). Thus, $q - p$ would have at least $n+1$ zeros. However since $q - p \in \mathbb{P}_n$, we must have $q = p$ which is a contradiction. So p is the best approximation to f out of $\mathbb{P}_n$. □

Theorem 4.14. *Among all polynomials in $\mathbb{P}_n$, $n \geq 1$ whose leading coefficient is 1, the polynomial*

$$p(x) = 2^{1-n} T_n(x)$$

is the one with the smallest L^∞ norm.

Proof. Any polynomial $e \in \mathbb{P}_n$ whose leading coefficient is 1, can be written as the difference between $f(x) = x^n$ and a polynomial $q \in \mathbb{P}_{n-1}$. By the previous theorem

$$\min_{q \in \mathbb{P}_n} \|f - q\|_\infty = \|2^{1-n} T_n\|_\infty.$$

Therefore among all polynomials in $\mathbb{P}_n$ whose leading coefficient is 1, the polynomial $p(x) = 2^{1-n} T_n(x)$ has the smallest deviation from zero. □

Theorem 4.15. *Let $A \in \mathbb{C}^{n \times n}$ be HPD and $f \in \mathbb{C}^n_\star$. The error for the zero-start conjugate gradient method satisfies*

$$\|e_k\|_A \leq 2 \left(\frac{\sqrt{k_2(A)} - 1}{\sqrt{k_2(A)} + 1} \right)^k \|e_0\|_A$$

Proof. Suppose that $\sigma(A) = \{\lambda_1, \ldots, \lambda_n\}$, $0 \leq \lambda_1 \leq \cdots \leq \lambda_n$. From a previous theorem

$$\|e_k\|_A \leq \max_{\lambda \in [\lambda_1, \lambda_n]} |q_k(\lambda)| \|e_0\|_A, \tag{4.36}$$

where q_k is a polynomial of degree at most k, such that $q_k(0) = 1$. Since this polynomial is arbitrary, we set

$$q_k(t) = \frac{1}{T_k(1/p)} T_k \left(\frac{1}{p} \left(1 - \frac{2}{\lambda_1 + \lambda_n} t \right) \right),$$

where $p = \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}$ and $T_k(x)$ is the Chebyshev polynomial. We obtain the bound

$$\begin{aligned}
\max_{\lambda \in [\lambda_1, \lambda_n]} |q_k(\lambda)| &= \max_{\lambda \in [\lambda_1, \lambda_n]} \left| \tfrac{1}{T_k(1/p)} T_k \left(\tfrac{1}{p} \left(1 - \tfrac{2}{\lambda_1 + \lambda_n} \lambda \right) \right) \right| \\
&\leq \left| \tfrac{1}{T_k(1/p)} \right| \max_{\lambda \in [\lambda_1, \lambda_n]} \left| T_k \left(\tfrac{1}{p} \left(1 - \tfrac{2}{\lambda_1 + \lambda_n} \lambda \right) \right) \right| \\
&\leq \left| \tfrac{1}{T_k(1/p)} \right| \left| T_k \left(\tfrac{1}{p} \left(\tfrac{\lambda_1 + \lambda_n - 2\lambda_1}{\lambda_1 + \lambda_n} \right) \right) \right| \\
&\leq \left| \tfrac{1}{T_k(1/p)} \right| |(-1)^k|.
\end{aligned}$$

Since $p < 1 \Rightarrow \frac{1}{p} > 1$, so we set $\frac{1}{p} = \cosh \sigma \Rightarrow \sigma = \cosh^{-1}\left(\frac{1}{p}\right)$. As a consequence,

$$T_k(1/p) = \cosh(k\sigma) = \frac{1}{2}(e^{k\sigma} + e^{-k\sigma}) \geq \frac{1}{2} e^{k\sigma}.$$

It holds that

$$\begin{aligned}
\sigma = \cosh^{-1}(1/p) \quad &\Rightarrow \quad \cosh \sigma = \frac{1}{p} \\
&\Rightarrow \quad \frac{1}{p} = \frac{e^{2\sigma} + 1}{2e^\sigma} \\
&\Rightarrow \quad pe^{2\sigma} - 2e^\sigma + p = 0.
\end{aligned}$$

Defining $\omega = e^\sigma$, we have

$$\omega = \frac{1}{p} + \frac{\sqrt{1 - p^2}}{p}.$$

Therefore,

$$\sigma = \ln\left(\frac{1}{p} + \sqrt{\frac{1}{p^2} - 1}\right),$$

and

$$e^{k\sigma} = \left(\frac{1}{p}(1+\sqrt{1-p^2})\right)^k.$$

Observe that

$$p = \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} = \frac{\lambda_n/\lambda_1 - 1}{\lambda_n/\lambda_1 + 1} = \frac{\kappa_2(A) - 1}{\kappa_2(A) + 1}.$$

Finally, according (4.4), we have the inequality

$$\begin{aligned}
\frac{1}{T_k(1/p)} \le 2(e^{-\sigma})^k &= 2\left(\frac{p}{1+\sqrt{1-p^2}}\right)^k \\
&= 2\left(\frac{\kappa_2(A)-1}{(\sqrt{\kappa_2(A)}+1)^2}\right)^k \\
&= 2\left(\frac{(\sqrt{\kappa_2(A)}-1)(\sqrt{\kappa_2(A)}+1)}{(\sqrt{\kappa_2(A)}+1)^2}\right)^k \\
&= 2\left(\frac{(\sqrt{\kappa_2(A)}-1)}{(\sqrt{\kappa_2(A)}+1)}\right)^k.
\end{aligned}$$

The desired inequality follows from the last relation and (4.36)

$$\|e_k\|_A \le \frac{1}{T_k(1/p)}\|e_0\|_A \le 2\left(\frac{(\sqrt{\kappa_2(A)}-1)}{(\sqrt{\kappa_2(A)}+1)}\right)^k \|e_0\|_A.$$

□

4.4.1 Non-zero starting values

In general, we can think of the case of non-zero starting values as a variation of CG method with zero starting values. By a transformation of problem's data we obtain an equivalent problem with zero-starting values and hence the previous technique can be applied.

Definition 4.10. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n_\star$ and $x = A^{-1}f$. The **conjugate gradient method** is an iterative scheme for producing a sequence of approximations $\{x_k\}_{k=1}^\infty$ where*

$$x_k = x'_k + x_0, \qquad x'_k = \underset{z\in\mathscr{K}_k}{argmin} E_A(z+x_0), \tag{4.37}$$

where $\mathscr{K}_k = \mathscr{K}_k(A, r_0)$ with $r_0 = f - Ax_0$ and starting point $x_0 \in \mathbb{C}^n$.

Proposition 4.6. *Suppose that $A \in \mathbb{C}^{n\times n}$ is HPD, $f \in \mathbb{C}^n_\star$, $x = A^{-1}f$, $x_0 \in \mathbb{C}^n$ and set*

$$x' = x - x_0, \qquad r_0 = f - Ax_0.$$

The sequence $\{x'_k\}_{k=1}^\infty$ generated by the zero-start conjugate gradient method is equivalent to the sequence $\{x_k\}_{k=1}^\infty$ generated by the conjugate gradient (4.37) with starting vector x_0 to approximate the solution to $Ax = f$ in the sense that

$$x_k = x_0 + x'_k \quad \text{and} \quad x - x_k = x' - x'_k.$$

Furthermore as long as $x_0 \neq x$ there is an integer $m_\star \in \{1,\dots,n\}$ such that

$$x_k \neq x, \quad k = 1,\dots,m_\star - 1 \quad \text{and} \quad x_k = x, \quad k \ge m_\star.$$

Proof. We have that

$$E_A(z) = \frac{1}{2}\|z\|_A^2 - \Re(z^H f), \qquad \forall z \in \mathbb{C}^n,$$

and define

$$\tilde{E}_A(z) = \frac{1}{2}\|z\|_A^2 - \Re(z^H r_0), \qquad \forall z \in \mathbb{C}^n,$$

so that

$$\begin{aligned}
E_A(z+x_0) &= \frac{1}{2}\|z+x_0\|_A^2 - \Re(x_0^H f + z^H f) \\
&= \frac{1}{2}\|z\|_A^2 + \frac{1}{2}\|x_0\|_A^2 + \frac{1}{2}z^H A x_0 + \frac{1}{2}x_0^H A z - \Re(x_0^H f) - \Re(z^H f) \\
&= E_A(x_0) + \frac{1}{2}\|z\|_A^2 - \Re(-z^H A x_0 + z^H f) \\
&= E_A(x_0) + \frac{1}{2}\|z\|_A^2 - \Re(z^H r_0) \\
&= E_A(x_0) + \tilde{E}_A(z).
\end{aligned}$$

By definition we conclude that

$$x_k' = \underset{z \in \mathscr{K}_k}{argmin} E_A(z+x_0) = \underset{z \in \mathscr{K}_k}{argmin} \tilde{E}_A(z)$$

□

Remark 4.3. *The standard conjugate gradient method to approximate the solution of $Ax = f$ is equivalent to the zero-start conjugate gradient method to approximate the solution of $Ax' = r_0$.*

4.4.2 Preconditioned conjugate gradient method

Proposition 4.7. *Suppose that $A, L \in \mathbb{C}^{n\times n}$ are HPD, $f \in \mathbb{C}_\star^n$, $x = A^{-1}f \in \mathbb{C}_\star^n$. Let $L = B^H B$, where $B \in \mathbb{C}^{n\times n}$ is invertible. Define*

$$C = B^{-H} A B^{-1}.$$

Then C is HPD and $L^{-1}A$ is similar to C,

$$C = P^{-1}(L^{-1}A)P.$$

Consequently,

$$\sigma(L^{-1}A) = \sigma(C) \subset (0, \infty).$$

The following are equivalent for $x \in \mathbb{C}^n$:

1. $Ax = f$
2. $L^{-1}Ax = L^{-1}f$
3. $Bx = y, \quad Cy = q$ *where* $q = B^{-H}f$

The equation $Cy = q$ is called the preconditioned system.

Proposition 4.8. *Suppose that B, C, L and q are defined as in previous proposition. The sequence generated by the zero-start conjugate gradient method to approximate the solution to the preconditioned system*

$$Cy = q,$$

denoted by $\{y_k\}_{k=1}^\infty$, is the same sequence as that generated by the following recursive algorithm:

$y_0 = 0, s_0 = q - Cy_0 = q, d_0 = s_0 = q$
for $k = 0, 1, \cdots$ **do**
  if $k \geq 1$ **then**
    $d_k = s_k + \nu_k d_{k-1}, \qquad \nu_k = (s_k, s_k)_2 \;/\; (s_{k-1}, s_{k-1})_2$
  end
  $y_{k+1} = y_k + \theta_{k+1} d_k, \qquad \theta_{k+1} = (s_k, s_k)_2 \;/\; (Cd_k, d_k)_2$
  $s_{k+1} = s_k - \theta_{k+1} Cd_k$
  if $d_{k+1} = 0$ **then**
    stop
  end
end

Corollary 4.3. *With the same assumptions and notation as in Proposition 4.8, define for* $0 \leq k \leq m_\star$, *the vectors*

$$\begin{aligned} x_k &= B^{-1} y_k \\ r_k &= B^H s_k \\ p_k &= B^{-1} d_k, \end{aligned}$$

which are generated by the following recursive algorithm:

$x_0 = 0, r_0 = f, p_0 = L^{-1} f$
for $k = 0, 1, \cdots$ **do**
  if $k \geq 1$ **then**
    $p_k = L^{-1} r_k - \nu_k p_{k-1}, \qquad \nu_k = (L^{-1} r_k, r_k)_2 \;/\; (L^{-1} r_{k-1}, r_{k-1})_2$
  end
  $x_{k+1} = x_k + \theta_{k+1} p_k, \qquad \theta_{k+1} = (L^{-1} r_k, r_k)_2 \;/\; (Ap_k, p_k)_2$
  $r_{k+1} = r_k - \theta_{k+1} Ap_k$
  if $p_{k+1} = 0$ **then**
    stop
  end
end

4.4.3 Krylov subspace methods for non HPD problems

In previous paragraphs we discussed solving linear systems of the form $Ax = f$, where A is an HPD matrix. Now we will study the case of a linear system, $Ax = f$, where A is a non HPD matrix.

So suppose the linear system where $A \in \mathbb{C}^{n \times n}$ is non HPD. Conjugate Gradient method is not applied to this system but it is applicable to the equivalent system

$$A^H A x = A^H f.$$

The last one is an HPD system and is known as the system of *normal equations* and that because

$$\begin{aligned} A^H A x = A^H f \quad &\Rightarrow \quad A^H (f - Ax) = 0 \\ &\Rightarrow \quad A^H r = 0. \end{aligned}$$

Thus, the system consists of normal equations. This approach is called *Conjugate Gradient Normal equation Residual (CGNR)* method.
The algorithm of CGNR has the following form:

$r_0 = f - Ax_0, \;\; z_0 = A^H r_0, \;\; p_0 = z_0.$
for $k = 0, 1, \cdots$ **do**
  $w_i = Ap_i$
  $\alpha_i = (z_i, z_i)_2 \;/\; (w_i, w_i)_2$
  $x_{i+1} = x_i + \alpha_{i+1} w_i$
  $r_{i+1} = r_i - \alpha_{i+1} w_i$
  $z_{i+1} = A^H r_{i+1}$
  $\beta_i = (z_{i+1}, z_{i+1})_2 \;/\; (z_i, z_i)_2$
  $p_{i+1} = z_{i+1} + \beta_i p_i$
end

Define now, the vector y as $x = A^H y$ so that the given system becomes

$$AA^H y = f, \tag{4.38}$$

which again is known as the system of normal equations. The application of the CG method to (4.38) is called *Conjugate Gradient Normal equation Error (CGNE)* method. The algorithm of CGNE method is:

$r_0 = f - Ax_0, \; z_0 = A^H r_0, \; p_0 = z_0.$
for $k = 0, 1, \cdots$ **do**
  $\alpha_i = (z_i, r_i)_2 \;/\; (p_i, p_i)_2$
  $w_i = A * p_i$
  $x_{i+1} = x_i + \alpha_{i+1} w_i$
  $r_{i+1} = r_i - \alpha_{i+1} w_i$
  $z_{i+1} = A^H r_{i+1}$
  $\beta_i = (z_{i+1}, r_{i+1})_2 \;/\; (z_i, r_i)_2$
  $p_{i+1} = A^H z_{i+1} + \beta_i p_i$
end

We will define the GMRES method which be used to find the $x_n \in \mathscr{K}_n$ that minimizes the residual $r_n = \|f - Ax_n\|_2$.

Definition 4.11. *Let $A \in \mathbb{C}^{n\times n}$ be nonsingular, $f \in \mathbb{C}^n_\star$ and $x_0 \in \mathbb{C}^n$ be arbitrary. The sequence $\{x_k\}_{k=1}^\infty$ obtained by minimizing*

$$\|r(z)\|_2 = \|f - Az\|_2$$

over

$$x_0 + \mathscr{K}_k(A, f - Ax_0),$$

gives rise to the ***Generalized Minimal Residual (GMRES)*** *method.*

We saw that in the Conjugate Gradient method, the residuals form an orthogonal basis for $\mathscr{K}_k(A, f) = \langle r_0, \cdots, r_{k-1}\rangle$. In contrast, in GMRES this basis formed by:

$w^{(i)} = Av^{(i)}$
for $k = 1, \ldots, i$ **do**
  $w^{(i)} = w^{(i)} - (w^i, v^{(k)})v^{(k)}$
end
$v^{(i+1)} = w^{(i)} / \|w^{(i)}\|$

This is a modified Gram-Schmidt orthogonalization and applied to the Krylov sequence $\{A^k r^{(0)}\}$ is called the *Arnoldi method*. The GMRES iterates are constructed as

$$x^{(i)} = x^{(0)} + y_1 v^{(1)} + \cdots + y_i v^{(i)},$$

where y_k minimizes the residual norm $\|f - Ax^{(i)}\|$.

Based on drawbacks of Conjugate Gradient method and GMRES method we introduce the **BiConjugate Gradient method (BiCG)**. We will start the presentation of BiCG, name the main drawbacks of these two methods. Firstly, CG method is suitable for HPD matrices. In contrast, GMRES is applicable to non HPD matrices however, is requires more storage because of the large amount of residual minimizations. BiCG method constructs four sequences of vectors. Two sequences of residuals, $r^{(i)}, \tilde{r}^{(i)}$ which define as

$$r^{(i)} = r^{(i-1)} - \alpha_i A p^{(i)}, \qquad \tilde{r^{(i)}} = \tilde{r}^{(i-1)} - \alpha_i A \tilde{p}^{(i)},$$

and two sequences of search directions $p^{(i)}, \tilde{p}^{(i)}$ which define as:

$$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}, \qquad \tilde{p}^{(i)} = \tilde{r}^{(i-1)} + \beta_{i-1} \tilde{p}^{(i-1)},$$

where α_i and β_i are given by

$$\alpha_i = \frac{\tilde{r}^{(i-1)^T} r^{(i-1)}}{\tilde{p}^{(i)^T} A p^{(i)}}, \qquad \beta_i = \frac{\tilde{r}^{(i)^T} r^{(i)}}{\tilde{r}^{(i-1)^T} r^{(i-1)}}.$$

These vectors ensure the bi-orthogonal relations

$$\tilde{r}^{(i)^T} r^{(j)} = \tilde{p}^{(i)^T} A p^{(j)}, \qquad \text{if } i \neq j.$$

BiCG method is the basis for some successful iterative methods. One of them is **BiCGSTAB** method for nonsymmetric linear systems developed by *Hendrik Albertus van der Vorst* (see [10]). BiCGSTAB has faster and smoother convergence than BiCG.

The unpreconditioned BiCGSTAB algorithm starts with an initial guess of x_0. The initial residual is given by $r_0 = b - Ax_0$ and we choose an arbitrary vector r_0, such that $(\hat{r}_0, r_0) \neq 0$, e.g., $\hat{r}_0 = r_0$. The algorithm is as below:

$x_0 =$ initial guess
$r_0 = b - Ax_0$
$\hat{r}_0$ an arbitrary vector such that $(\hat{r}_0, r_0) \neq 0$
$\rho_0 = \alpha = \omega_0 = 1$
$v_0 = p_0 = 0$
for $i = 1, 2, 3, \cdots$ **do**
  $\rho_i = (\hat{r}_0, r_{i-1}); \beta = (\rho_i/\rho_{i-1})(\alpha/\omega_{i-1})$
  $p_i = r_{i-1} + \beta(p_{i-1} - \omega_{i-1} v_{i-1})$
  $v_i = Ap_i$
  $\alpha = \rho_i/(\hat{r}_0, v_i)$
  $s = r_{i-1} - \alpha v_i$
  $t = As$
  $\omega_i = (t,s)/(t,t)$
  $x_i = x_{i-1} + \alpha p_i + \omega_i s$
  if x_i *is accurate enough* **then**
    quit
  end
  $r_i = s - \omega_i t$
end

CHAPTER 5

The Helmholtz Equation

5.1 The Helmholtz equation

The study of many physical problems related to steady–state oscillations leads to the *Helmholtz equation*, named after Hermann von Helmholtz (1821–1894). It has various applications in physics, including optics, acoustics, electrostatics and quantum mechanics. In this work we concentrate on the two-dimensional Helmholtz equation

$$\Delta u + k^2 n(x,y) u = 0, \tag{5.1}$$

as a model of sound propagation. Equation (5.1) with a suitable radiation condition at infinity describes the propagation and scattering of time–harmonic waves. Here, u represents the acoustic pressure, $n(x,y)$ the index of refraction, and k the wave number, equal to $2\pi f/c_0$, where f is the frequency and c_0 is some reference sound speed. We will consider (5.1) in a finite rectangular waveguide Ω and assume the so-called *Sommerfeld radiation condition*

$$\lim_{r\to\infty} r^{1/2}\left(\frac{\partial u}{\partial \nu} - iku\right) = 0 \quad \text{on } \partial\Omega. \tag{5.2}$$

Discretizations of the Helmholtz problem (5.1) using, say, finite difference methods, result in a linear system

$$Au = b,$$

where A is a complex, non-Hermitian, indefinite, large, sparse matrix. This complex linear system is usually solved by a Gauss elimination type direct method, or by Krylov subspace methods like the quasi–minimal residual (QMR) or bi-conjugate gradient methods. Some Krylov subspace methods are used on the equivalent real system of double size. We consider the model problem, see [9]

$$-\Delta u - a^2(x,y)u + iq^2 u = f \quad \text{in } \Omega = (0,1)^2, \tag{5.3}$$

$$\frac{\partial u}{\partial \nu} + i\alpha u = 0 \quad \text{on } \Gamma \equiv \partial\Omega, \tag{5.4}$$

where q and α are real constants and a is a real-valued, bounded and sufficiently smooth function. We assume that this problem has a unique and sufficiently smooth solution in $H^1(\Omega)$.

5.2 A finite difference method for the model problem

In the model problem for the Helmholtz equation (5.3)–(5.4), we split the exact solution u and the right-hand side f into their real u_R, f_R and imaginary parts u_I, f_I,respectively. It is easily seen that (5.3) is equivalent to the following two problems

$$-\Delta u_R - a^2(x,y)u_R = f_R + q^2 u_I \quad \text{in } \Omega, \tag{5.5}$$

$$\frac{\partial u_R}{\partial v} = \alpha u_I \quad \text{on } \partial\Omega, \tag{5.6}$$

and

$$-\Delta u_I - a^2 u_I = f_I + q^2 u_R \quad \text{in } \Omega, \tag{5.7}$$

$$\frac{\partial u_I}{\partial v} = -\alpha u_R \quad \text{on } \partial\Omega. \tag{5.8}$$

Consider a regular partition of the Ω

$$(x_i, y_j) = (ih, jh), \qquad i, j = 0, 1, \ldots, N+1,$$

where $h = \frac{1}{N+1}$ is the step size and $N > 1$, integer, is the number of the interior points along each direction. We discretize the equation (5.3) using a second order, central difference scheme. As discussed in Section 2.1, using Taylor's theorem we have

$$u(x-h,y) \approx u(x,y) - h\frac{\partial u}{\partial x} + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}, \tag{5.9}$$

$$u(x+h,y) \approx u(x,y) + h\frac{\partial u}{\partial x} + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}, \tag{5.10}$$

$$u(x,y-h) \approx u(x,y) - h\frac{\partial u}{\partial y} + \frac{h^2}{2}\frac{\partial^2 u}{\partial y^2}, \tag{5.11}$$

$$u(x,y+h) \approx u(x,y) + h\frac{\partial u}{\partial y} + \frac{h^2}{2}\frac{\partial^2 u}{\partial y^2}. \tag{5.12}$$

Adding (5.9) and (5.10), and then (5.11) and (5.12) we obtain the central difference approximations

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \approx \frac{1}{h^2}[u_{i+1,j} + u_{i-1,j} - 2u_{i,j}],$$
$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) \approx \frac{1}{h^2}[u_{i,j+1} + u_{i,j-1} - 2u_{i,j}],$$

where $u_{k,l} \approx u(x_k, y_l)$, $0 \le k, l \le N+1$. Using these approximations in (5.3) we obtain the finite difference scheme

$$-\frac{1}{h^2}[u_{i+1,j} + u_{i-1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}] - a_{i,j}^2 u_{i,j} + iq^2 u_{i,j} = f_{i,j}, \qquad 0 \le i, j \le N+1,$$

where $a_{i,j} = a(x_i, y_j)$. We let $g_{i,j} = 4 + h^2(iq^2 - a_{i,j}^2)$ and $b_{i,j} = h^2 f_{i,j}$, so that the finite difference scheme may be written as

$$-u_{i+1,j} - u_{i-1,j} - u_{i,j-1} - u_{i,j+1} + g_{i,j}u_{i,j} = b_{i,j}, \qquad 0 \le i, j \le N+1. \tag{5.13}$$

The discretization of the boundary condition (5.4) is as follows:

1. On the top edge the unit outward normal is $\vec{v}_1 = (0,1)$ and the boundary condition reduces to

$$\frac{\partial u}{\partial y} + i\alpha u = 0,$$

which we discretize using the central difference formula

$$\frac{u_{i,N+2}-u_{i,N}}{2h}+i\alpha u_{i,N+1}=0, \qquad 1\le i\le N.$$

Now, for $j=N+1$ and $1\le i\le N$, we have from (5.13) that

$$-u_{i+1,N+1}-u_{i-1,N+1}-u_{i,N}-u_{i,N+2}+g_{i,N+1}u_{i,N+1}=b_{i,N+1}, \qquad 1\le i\le N.$$

Eliminating the fictitious term $u_{i,N+2}$ we obtain the equation

$$-u_{i+1,N+1}-u_{i-1,N+1}-2u_{i,N}+(g_{i,N+1}+i2h\alpha)u_{i,N+1}=b_{i,N+1}, \qquad 1\le i\le N. \tag{5.14}$$

2. On the bottom edge the unit outward normal $\vec{v}_2=(0,-1)$ and the boundary condition reduces to

$$-\frac{\partial u}{\partial y}+i\alpha u=0, \qquad 1\le i\le N,$$

which we discretize using the central difference formula

$$-\frac{u_{i,1}-u_{i,-1}}{2h}+i\alpha u_{i,0}, \qquad 1\le i\le N.$$

Now, for $j=0$ and $1\le i\le N$, we have from (5.13) that

$$-u_{i+1,0}-u_{i-1,0}-u_{i,-1}-u_{i,1}+g_{i,0}u_{i,0}=b_{i,0}, \qquad 1\le i\le N.$$

Eliminating the fictitious term $u_{i,-1}$ we obtain the equation

$$-u_{i+1,0}-u_{i-1,0}-2u_{i,1}+(g_{i,0}+i2h\alpha)u_{i,0}=b_{i,0}, \qquad 1\le i\le N. \tag{5.15}$$

3. On the right edge the unit outward normal $\vec{v}_3=(1,0)$ and the boundary condition reduces to

$$\frac{\partial u}{\partial x}+i\alpha u=0,$$

which we discretize using the central difference formula

$$\frac{u_{N+2,j}-u_{N,j}}{2h}+i\alpha u_{N+1,j}=0, \qquad 1\le j\le N.$$

Now, for $i=N+1$ and $1\le j\le N$, we have from (5.13) that

$$-u_{N+2,j}-u_{N,j}-u_{N+1,j-1}-u_{N+1,j+1}+g_{N+1,j}u_{N+1,j}=b_{N+1,j}, \qquad 1\le j\le N.$$

Eliminating the fictitious term $u_{N+2,j}$ we obtain the equation

$$-u_{N+1,j-1}-2u_{N,j}-u_{N+1,j+1}+(g_{N+1,j}+i2h\alpha)u_{N+1,j}=b_{N+1,j}, \qquad 1\le j\le N. \tag{5.16}$$

4. On the left edge the unit outward normal $\vec{v}_4=(-1,0)$ and the boundary condition reduces to

$$-\frac{\partial u}{\partial x}+i\alpha u=0,$$

which we discretize using the central difference formula

$$-\frac{u_{1,j}-u_{-1,j}}{2h}+i\alpha u_{0,j}=0, \qquad 1\le j\le N.$$

Now, for $i=0$ and $1\le j\le N$, we have from (5.13) that

$$-u_{1,j}-u_{-1,j}-u_{0,j-1}-u_{0,j+1}+g_{0,j}u_{0,j}=b_{0,j}, \qquad 1\le j\le N.$$

Eliminating the fictitious term $u_{-1,j}$ we obtain the equation

$$-u_{0,j-1}-2u_{1,j}-u_{0,j+1}+(g_{0,j}+i2h\alpha)u_{0,j}=b_{0,j}, \qquad 1\le j\le N. \tag{5.17}$$

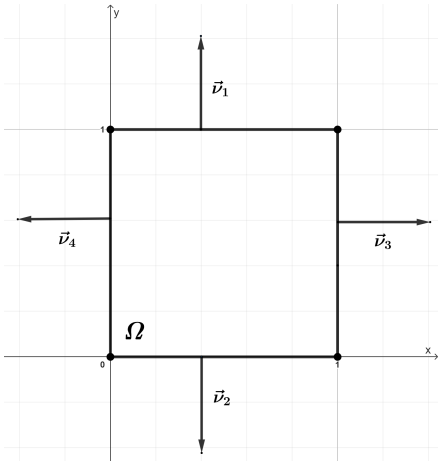

Figure 5.1: Graph of the domain $\Omega = (0,1) \times (0,1)$ and the corresponding outward unit normal vectors.

Equations (5.13), for $1 \le i, j \le N$, and (5.14)–(5.17) comprise a $(N+2)^2 \times (N+2)^2$ system of linear equations of the form

$$\mathcal{A}u = b, \tag{5.18}$$

where

$$u = [u_{0,0}, u_{1,0}, \dots, u_{N+1,0}, u_{0,1}, u_{1,1}, \dots, u_{N+1,1}, \dots, u_{0,N+1}, u_{1,N+1}, \dots u_{N+1,N+1}]^T,$$
$$b = [b_{0,0}, b_{1,0}, \dots, b_{N+1,0}, b_{0,1}, b_{1,1}, \dots, b_{N+1,1}, \dots, b_{0,N+1}, b_{1,N+1}, \dots b_{N+1,N+1}]^T,$$

and

$$\mathcal{A} = \begin{bmatrix} \mathcal{D}_0 & -2I & & & \\ -I & \mathcal{D}_1 & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & \mathcal{D}_N & -I \\ & & & -2I & \mathcal{D}_{N+1} \end{bmatrix} \in \mathbb{C}^{(N+2)^2 \times (N+2)^2}.$$

Here, $I \in \mathbb{R}^{(N+2)\times(N+2)}$ is the identity matrix, and

$$\mathcal{D}_j = \begin{cases} \hat{\mathcal{D}}_j & \text{for } j = 1, \dots, N, \\ \hat{\mathcal{D}}_j + i2h\alpha I & \text{for } j = 0, N+1, \end{cases}$$

with

$$\hat{\mathcal{D}}_j = \begin{bmatrix} g_{0,j} + i2h\alpha & -2 & & & \\ -1 & g_{1,j} & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & g_{N,j} & -1 \\ & & & -2 & g_{N+1,j} + i2h\alpha \end{bmatrix}, \qquad j = 0, \dots, N+1.$$

Obviously the matrix $\mathcal{A}$ is a non symmetric matrix. We can symmetrize it as follows: define the diagonal matrix

$$J = \operatorname{diag}\left(\frac{1}{\sqrt{2}}, 1, \dots, 1, \frac{1}{\sqrt{2}}\right) \in \mathbb{R}^{(N+2)\times(N+2)}, \tag{5.19}$$

and set $S = J \otimes J$ where $\otimes$ denotes the Kronecker product.

Definition 5.1. *If $\boldsymbol{A}$ is a $m \times n$ matrix and $\boldsymbol{B}$ is a $q \times p$ matrix, then the Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B}$ is the $pm \times qn$ block matrix:*

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} \alpha_{11}\boldsymbol{B} & \cdots & \alpha_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ \alpha_{m1}\boldsymbol{B} & \cdots & \alpha_{mn}\boldsymbol{B} \end{bmatrix}$$

Set $S = J \otimes J$ and consider the system

$$Av = c, \tag{5.20}$$

where $A = S\mathcal{A}S^{-1}$, $v = Su$, $c = Sb$, so that

$$A = \begin{bmatrix} D_0 & -\sqrt{2}I & & & & \\ -\sqrt{2}I & D_1 & -I & & & \\ & -I & D_2 & -I & & \\ & & \ddots & \ddots & \ddots & \\ & & & -I & D_N & -\sqrt{2}I \\ & & & & -\sqrt{2}I & D_{N+1} \end{bmatrix}, \tag{5.21}$$

$$D_j = \begin{cases} \hat{D}_j & \text{for } j = 1, \ldots, N, \\ \hat{D}_j + i2h\alpha I, & \text{for } j = 0, N+1, \end{cases} \tag{5.22}$$

with

$$\hat{D}_j = \begin{bmatrix} g_{0,j} + 2h\alpha i & -\sqrt{2} & & & \\ -\sqrt{2} & g_{1,j} & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & g_{N,j} & -\sqrt{2} \\ & & & -\sqrt{2} & g_{N+1,j} + 2h\alpha i \end{bmatrix}, \qquad j = 0, \cdots, N+1. \tag{5.23}$$

It is easy to see that the system $Av = c$ is equivalent to (5.18) , with A a complex symmetric matrix. We now split A, v and c in real and imaginary parts

$$A = A_1 + iA_2, \quad v = v_{\mathbb{R}} + iv_{\mathbb{C}}, \quad c = c_{\mathbb{R}} + ic_{\mathbb{C}},$$

and transform the complex linear system (5.20) into the following equivalent real, block system

$$\tilde{A}\tilde{v} = \tilde{c}, \tag{5.24}$$

where

$$\tilde{v} = \begin{bmatrix} v_{\mathbb{R}} \\ v_{\mathbb{C}} \end{bmatrix} \in \mathbb{R}^{2(N+2)^2}, \qquad \tilde{c} = \begin{bmatrix} c_{\mathbb{R}} \\ c_{\mathbb{C}} \end{bmatrix} \in \mathbb{R}^{2(N+2)^2},$$

$$\tilde{A} = \begin{bmatrix} A_2 & A_1 \\ A_1 & -A_2 \end{bmatrix} \in \mathbb{R}^{2(N+2)^2 \times 2(N+2)^2}, \tag{5.25}$$

$$A_1 = \begin{bmatrix} B_0 & -\sqrt{2}I & & & & \\ -\sqrt{2}I & B_1 & -I & & & \\ & -I & B_2 & -I & & \\ & & \ddots & \ddots & \ddots & \\ & & & -I & B_N & -\sqrt{2}I \\ & & & & -\sqrt{2}I & B_{N+1} \end{bmatrix} \in \mathbb{R}^{(N+2)^2 \times (N+2)^2}, \tag{5.26}$$

$$A_2 = \operatorname{diag}(C_0, C_1, \cdots, C_1, C_0) \in \mathbb{R}^{(N+2)^2 \times (N+2)^2}, \tag{5.27}$$

with

$$C_0 = \operatorname{diag}(2h\alpha, 0, \ldots, 0, 2h\alpha) + (2h\alpha + h^2q^2)I, \tag{5.28}$$

$$C_1 = \operatorname{diag}(2h\alpha, 0, \ldots, 0, 2h\alpha) + h^2q^2I, \tag{5.29}$$

and

$$B_j = \begin{bmatrix} b_{0,j} & -\sqrt{2} & & & & \\ -\sqrt{2} & b_{1,j} & -1 & & & \\ & -1 & b_{2,j} & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & b_{N,j} & -\sqrt{2} \\ & & & & -\sqrt{2} & b_{N+1,j} \end{bmatrix},$$

where $b_{i,j} = 4 - h^2\alpha_{i,j}^2$.

We solve the real system (5.24) using two variations of the block SOR method defined by the iteration matrix

$$T_{SOR} = (L + \omega^{-1}D)^{-1}[(\omega^{-1} - 1)D - U], \tag{5.30}$$

the first one denoted by $BSOR1$, where

$$D_1 = \begin{bmatrix} A_2 & 0 \\ 0 & -A_2 \end{bmatrix}, \quad L_1 = \begin{bmatrix} 0 & 0 \\ A_1 & 0 \end{bmatrix}, \quad U_1 = \begin{bmatrix} 0 & A_1 \\ 0 & 0 \end{bmatrix} \tag{5.31}$$

with corresponding iteration matrix

$$T_{BSOR1} = \begin{bmatrix} (1-\omega)I & -A_2^{-1}A_1 \\ \omega(1-\omega)A_2^{-1}A_1 & -\omega A_2^{-1}A_1A_2^{-1}A_1 + (1-\omega)I \end{bmatrix} \tag{5.32}$$

and the other one denoted by $BSOR2$ where

$$D_2 = \begin{bmatrix} A_1 & 0 \\ 0 & A_1 \end{bmatrix}, \quad L_2 = \begin{bmatrix} 0 & 0 \\ -A_2 & 0 \end{bmatrix}, \quad U_2 = \begin{bmatrix} 0 & A_2 \\ 0 & 0 \end{bmatrix} \tag{5.33}$$

with corresponding iteration matrix

$$T_{BSOR2} = \begin{bmatrix} (1-\omega)I & A_1^{-1}A_2 \\ -\omega(1-\omega)A_1^{-1}A_2 & -\omega A_1^{-1}A_2A_1^{-1}A_2 + (1-\omega)I \end{bmatrix}. \tag{5.34}$$

Thus, for these iteration matrices, we have the iterative schemes

$$\begin{pmatrix} v_{\mathbb{R}} \\ v_{\mathbb{C}} \end{pmatrix}^{(n+1)} = T_{BSOR1}\begin{pmatrix} v_{\mathbb{R}} \\ v_{\mathbb{C}} \end{pmatrix}^{(n)} + (L + \omega^{-1}D)^{-1}\begin{pmatrix} c_{\mathbb{C}} \\ v_{\mathbb{R}} \end{pmatrix}^{(n+1)}$$

for $BSOR1$ and

$$\begin{pmatrix} v_{\mathbb{R}} \\ v_{\mathbb{C}} \end{pmatrix}^{(n+1)} = T_{BSOR2}\begin{pmatrix} v_{\mathbb{R}} \\ v_{\mathbb{C}} \end{pmatrix}^{(n)} + (L + \omega^{-1}D)^{-1}\begin{pmatrix} c_{\mathbb{R}} \\ v_{\mathbb{C}} \end{pmatrix}^{(n+1)}$$

for $BSOR2$. These schemes may be written as

- BSOR1

$$\begin{aligned} v_{\mathbb{R}}^{(n+1)} &= (1-\omega)v_{\mathbb{R}}^{(n)} - A_2^{-1}A_1v_{\mathbb{C}}^{(n)} + \omega A_2^{-1}c_{\mathbb{C}} \\ v_{\mathbb{C}}^{(n+1)} &= \omega(1-\omega)A_2^{-1}A_1v_{\mathbb{R}}^{(n)} + \left[(1-\omega)I - \omega A_2^{-1}A_1A_2^{-1}A_1\right]v_{\mathbb{C}}^{(n)} + \omega^2A_2^{-1}A_1A_2^{-1}c_{\mathbb{C}} - \omega A_2^{-1}c_{\mathbb{R}} \end{aligned}$$

- BSOR2

$$\begin{aligned} v_{\mathbb{R}}^{(n+1)} &= (1-\omega)v_{\mathbb{R}}^{(n)} + A_1^{-1}A_2v_{\mathbb{C}}^{(n)} + \omega A_1^{-1}c_{\mathbb{R}} \\ v_{\mathbb{C}}^{(n+1)} &= -\omega(1-\omega)A_1^{-1}A_2v_{\mathbb{R}}^{(n)} + \left[(1-\omega)I - \omega A_1^{-1}A_2A_1^{-1}A_2\right]v_{\mathbb{C}}^{(n)} - \omega^2A_1^{-1}A_2A_1^{-1}c_{\mathbb{R}} + \omega A_1^{-1}c_{\mathbb{C}} \end{aligned}$$

and organize their computations as follows: In the first scheme, we multiply $v_{\mathbb{R}}^{(n+1)}$ by $\omega A_2^{-1}A_1$ and substitute the term $\omega^2 A_2^{-1}A_1A_2^{-1}c_{\mathbb{C}}$ into the formula for $v_{\mathbb{C}}^{(n+1)}$. For the second scheme, we multiply the $v_{\mathbb{R}}^{(n+1)}$ by $-\omega A_1^{-1}A_2$ and substitute the term $-\omega^2 A_1^{-1}A_2A_1^{-1}c_{\mathbb{R}}$ into the formula for $v_{\mathbb{C}}^{(n+1)}$. Thus,

- BSOR1

$$\begin{aligned} v_{\mathbb{R}}^{(n+1)} &= (1-\omega)v_{\mathbb{R}}^{(n)} - \omega A_2^{-1}A_1 v_{\mathbb{C}}^{(n)} + \omega A_2^{-1}c_{\mathbb{C}}, \\ v_{\mathbb{C}}^{(n+1)} &= \omega A_2^{-1}A_1 v_{\mathbb{R}}^{(n+1)} + (1-\omega)v_{\mathbb{C}}^{(n)} - \omega A_2^{-1}c_{\mathbb{R}}, \end{aligned}$$

- BSOR2

$$\begin{aligned} v_{\mathbb{R}}^{(n+1)} &= (1-\omega)v_{\mathbb{R}}^{(n)} + \omega A_1^{-1}A_2 v_{\mathbb{C}}^{(n)} + \omega A_1^{-1}c_{\mathbb{R}}, \\ v_{\mathbb{C}}^{(n+1)} &= -\omega A_1^{-1}A_2 v_{\mathbb{R}}^{(n+1)} + (1-\omega)v_{\mathbb{C}}^{(n)} + \omega A_1^{-1}c_{\mathbb{C}}, \end{aligned}$$

where ω is the relaxation parameter of the SOR method.

5.3 A finite element method for the model problem

We consider equation (5.3) posed in a two-dimensional waveguide consisting of a single water layer overlying an acoustically-soft bottom of variable topography. The upper boundary is assumed to be a horizontal pressure-release surface. The acoustic field is generated by a time-harmonic source located at (x_s, y_s), see Figure 5.2.

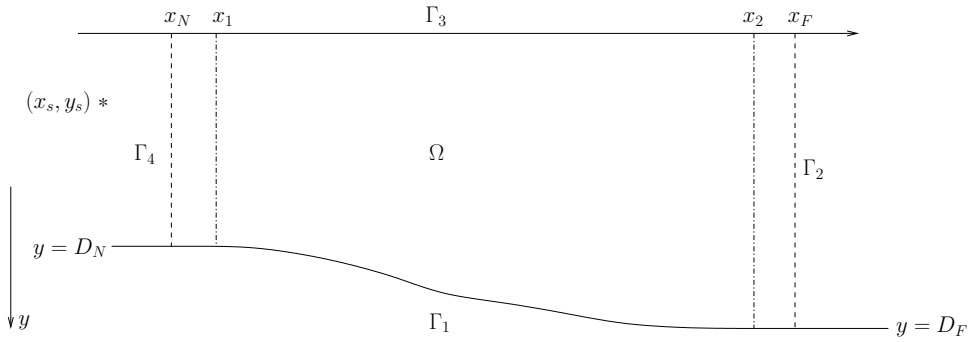

Figure 5.2: Schematic represenatation of the waveguide.

We take $q = f = 0$ and a, the real wavenumber, as a sufficiently smooth function of the form

$$a(x,y) = \begin{cases} a_N & \text{for } x \le x_1, \\ a_{\text{int}}(x,y) & \text{for } x_1 < x < x_2, \\ a_F & \text{for } x \le x_2. \end{cases}$$

Equation (5.3) is supplemented by appropriate radiation conditions, see [5] for details. The weak formulation of the boundary value problem is descritized by a finite element method with continuous in Ω piecewise linear functions. A regular triangulation with triangles of maximal diameter h is imposed on Ω. In the numerical experiments reported below we took $\Omega = [170, 220] \times [0, 50]$ and a triangulation consisting of 4671 elements.

Denote the resulting linear system of equations by $Ax = b$. The matrix A is complex, symmetric but not Hermitian. Its condition number was approximately 1.8×10^7, necessitating the use of a preconditioner. In the present case, we applied a preconditioning matrix based on the *LU factorization*, more specifically a sparse matrix resulting from an *incomplete LU factorization*. In contrast with the complete LU

factorization, in the incomplete LU factorization (ILU) computes a sparse lower triangular matrix L and a sparse upper triangular matrix U such that the residual matrix $R = LU - A$ satisfies certain constraints. Here, we discuss the ILU(0) factorization, the simplest form of the family of ILU preconditioners. ILU(0) is the incomplete LU factorization technique with no fill–in[1], and consists of taking the zero pattern P to be precisely the zero pattern of A. Consider the 5–point [2] matrix A, any lower and upper triangular

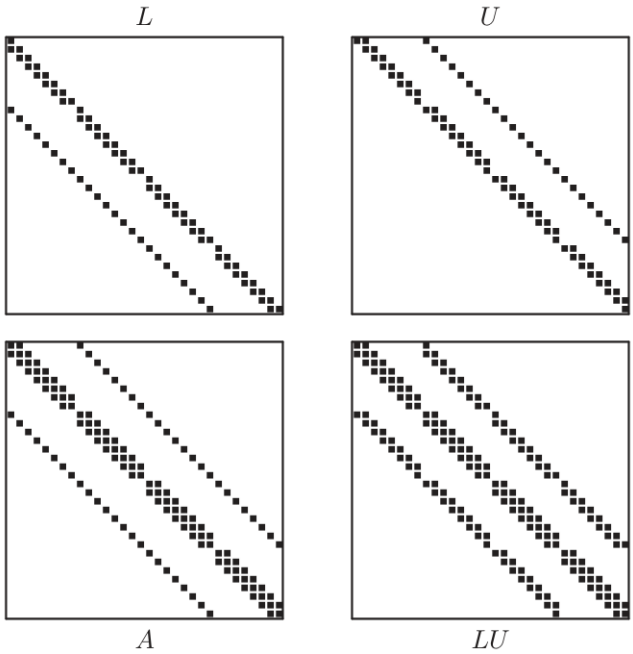

Figure 5.3: The ILU(0) factorization for a five–point matrix as it shown in [6]

matrix L and U which have the same structure as the lower and upper part of A, respectively, as it shows in Figure 5.3. The resulting matrix of the product LU would have the pattern shown at the bottom-left corner of Figure 5.3. The structure of LU is almost the same as the structure of A except of two extra diagonals which appear with nonzero elements. These extra diagonals are called *fill–in elements*. If these fill–in elements are ignored, then it is possible to find L and U so that their product is equal to A in the other diagonals. This is the ILU(0) preconditioner, with 0 representing the number of extra diagonals which are not ignored. Similarly, we may define ILU(p) incomplete factorizations which differ from ILU(0) in keeping the "pth–order fill–ins".

Now we apply some iterative methods to solve the system resulting from the finite element discretization of the Helmholtz equation using the finite element method. We refer to the python code in Appendix A for the details of the solution process. Lines $7-32$ contain the Python code necessary for reading the elements of the system matrix and its sparsisty pattern. In short,

- A is a complex array contains the complex values α_{ij} stored row by row, from row 1 to n.
- JA is an integer array contains the column indices of the elements α_{ij} as stored in the array A
- IA is an integer array contains the pointers to the beginning of each row in the arrays A and JA.

Our main purpose is to apply the GMRES and BiCGSTAB iterative methods to this system. However, using the python-supplied functions for GMRES and BICGSTAB we need to transform A to csc form. So, at lines $38-43$ we compute a new array *Rows* which contains the corresponding row of each element and then we transform A to a csc form. In lines $46-48$ we compute the preconditioning matrix. Finally, for

[1] *fill–in*: the entries of a matrix change from zero to a non–zero value
[2] for the definition see section 2.2.2 of [6]

comparison purposes, we also use the direct method SuperLU and compare the outputs. The following table shows the needed average time of each method:

Table 5.1: Comparison of GMRES, BiCGStab and SuperLU

	GMRES	BiCGStab	SuperLU
Convergence	True	True	True
Average Time	0.0011098	0.0004747	0.0024724

Observe that BiCGStab converges faster than the others even though the difference between them isn't as big as it could be.

Bibliography

[1] Richard Barrett et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. 1994. DOI: 10.1137/1.9781611971538. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611971538. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611971538.

[2] O. Buneman. Compact non-iterative poisson solver. In: (Jan. 1969). URL: https://www.osti.gov/biblio/4771228.

[3] Timothy A. Davis and Yifan Hu. The University of Florida Sparse Matrix Collection. ACM Transactions on Mathematical Software 38,1, Article 1. In: (2011). DOI: 10.1145/2049662.2049663. URL: https://doi.org/10.1145/2049662.2049663.

[4] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. 3rd ed. The Johns Hopkins University Press, 1996.

[5] D.A. Mitsoudis, Ch. Makridakis, and M. Plexousakis. Helmholtz equation with artificial boundary conditions in a two-dimensional waveguide. In: *SIAM Journal on Mathematical Analysis* 44 (2012), pp. 4320–4344.

[6] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718003. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898718003.

[7] Gilbert Strang. Multiplying and Factoring Matrices. In: *Am. Math. Mon.* 125.3 (2018), pp. 223–230.

[8] Roland A. Sweet. A Cyclic Reduction Algorithm for Solving Block Tridiagonal Systems of Arbitrary Dimension. In: *SIAM Journal on Numerical Analysis* 14.4 (1977), pp. 706–720. ISSN: 00361429. URL: http://www.jstor.org/stable/2156489 (visited on 12/08/2022).

[9] Manolis Vavalis. Real valued block SOR iterative methods for the Helmholtz equation. In: *International Journal of Numerical Methods and Applications* 10 (Jan. 2013).

[10] H. A. van der Vorst. Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. In: *SIAM Journal on Scientific and Statistical Computing* 13.2 (1992), pp. 631–644. DOI: 10.1137/0913035. eprint: https://doi.org/10.1137/0913035. URL: https://doi.org/10.1137/0913035.

[11] Γ.Δ. Ακρίβης και Β.Α. Δουγαλής. *Εισαγωγή στην Αριθμητική Ανάλυση*. 4η έκδοση. Πανεπιστημιακές Εκδόσεις Κρήτης, 2015.

APPENDIX A

First appendix

Python code for "upwind scheme"

```python

import numpy as np
from scipy.linalg import solve_banded
import matplotlib.pyplot as plt

a = 2; b = 9; N = 30
h = 1/(N+1)
x = np.linspace(0, 1, N+2)

c1=(b*h)/(2*a)
c2=(b*h)/a

Ab = np.zeros((3,N))
Z =np.zeros(N)
Z[N-1]=1

if (np.sign(b)==1):
  Ab[0, 1: ] =  (-1)*np.ones(N-1)
  Ab[1, : ] =  (2 + c2)*np.ones(N)
  Ab[2, :N-1] =  (-1-c2)*np.ones(N-1)
else:
  Ab[0, 1: ] =  (-1+c1)*np.ones(N-1)
  Ab[1, : ] =  2*np.ones(N)
  Ab[2, :N-1] =  (-1-c1)*np.ones(N-1)


sol = solve_banded((1,1), Ab, Z)
S = np.zeros(N+2)
S[1:N+1] = sol
S[N+1]=1

plt.plot(x, S)
plt.show()
```

Listing A.1: Upwind Scheme

Python code for the solution of a system with the CSR format of A

```
import matplotlib.pyplot as plt
import scipy.io as sio
import scipy.sparse
import numpy as np
from scipy.sparse.linalg import spsolve
import time
from scipy.sparse import csr_matrix

N_list=['bcsstk25.mat','bcsstk16.mat', 'bcsstk17.mat', 'wathen100.mat', 'bcsstk36.mat']
Second_method = np.ones(len(N_list))
Nonzero = np.ones(len(N_list))
Second_method2 = np.ones(len(N_list))

for i in range(0,len(N_list)):
  mat=scipy.io.loadmat(N_list[i])

  if i==3:
    print('Matrix:', N_list[i])
    A = mat['Problem'][0][0][2]

    plt.spy(mat['Problem'][0][0][2],precision = 0.2, markersize = 0.2)
    plt.show()

    N=mat['Problem'][0][0][2].size
    sum_of_rows = np.size(A, axis = 0)
    b=np.ones(sum_of_rows)

    T=scipy.sparse.find(A)
    K=np.size(T ,axis = 1)
    Nonzero[i]=K
    print('Non-zero elements:',K)

    B= csr_matrix(A)

    sum_of_cols = np.size(B, axis = 1)
    b=np.ones(sum_of_cols)

    k=time.perf_counter()
    x2= spsolve(B, b)
    t=time.perf_counter()

    print('Solve in ', t-k, ' seconds')
    print(x2)
    Second_method[i] = t-k

    print()
  else:
    print('Matrix:', N_list[i])
    A = mat['Problem'][0][0][1]
    plt.spy(mat['Problem'][0][0][1],precision = 0.2, markersize = 0.2)
    plt.show()

    N=mat['Problem'][0][0][1].size
    sum_of_rows = np.size(A, axis = 0)
    b=np.ones(sum_of_rows)

    T=scipy.sparse.find(A)
    K=np.size(T ,axis = 1)
    Nonzero[i]=K

    print('Non-zero elements:',K)
    B= csr_matrix(A)
```

```
    sum_of_cols = np.size(B, axis = 1)
    b=np.ones(sum_of_cols)

    k=time.perf_counter()
    x2= spsolve(B, b)
    t=time.perf_counter()

    print('Solve in ', t-k, ' seconds')
    print(x2)
    Second_method[i] = t-k

    print()
    print('Next matrix:')

new_unit2 = Second_method[0]
for i in range(0,len(N_list)):
temp=Second_method[i]
Second_method2[i]=temp/new_unit2

axis_x=[252241, 290378, 428650, 471601, 11443140]
default_x_ticks = range(len(axis_x))
axis_y2=[Second_method2[0], Second_method2[1], Second_method2[2], Second_method2[3],
  Second_method2[4]]
axis_y=[Second_method[0], Second_method[1], Second_method[2], Second_method[3], Second_method
  [4]]

fig, ax = plt.subplots()
ax.plot(default_x_ticks, axis_y)
plt.xticks(default_x_ticks, axis_x)
ax.set_xlabel('Non-zero elements')
ax.set_ylabel('Time (seconds)')

ax2 = ax.twinx()
ax2.plot(default_x_ticks, axis_y2)
ax2.set_ylabel('Normalized time')

plt.show()
```

Listing A.2: Solve in CSR format

Python code for Jacobi's method

```
import scipy.io as sio
import scipy.sparse
import numpy as np
from numpy import linalg as LA
import matplotlib.pyplot as plt
import time

#Define a matrix A
A = mat['Problem'][0][0][1]
n, nc = A.shape
b = np.ones(n)

# Stop if || b - A x_k || < tol
tol = 2.0e-4

# Allow at most maxiters iterations
maxiters = 4000

# Initial guess is x = 0
x = np.zeros(n)
xnew = np.zeros(n)
resvec = np.zeros(maxiters)
N = 0

tbeg = time.perf_counter()
while N < maxiters:
  res = np.linalg.norm(b - A.dot(x))
  resvec[N] = res
  for i in range(n):
    xnew[i] = b[i]
    for j in range(i)    : xnew[i] -= A[i,j] * x[j]
    for j in range(i+1,n): xnew[i] -= A[i,j] * x[j]
    xnew[i] /= A[i,i]

  if LA.norm(b - A @ xnew) < tol:
    break

  for i in range(n): x[i] = xnew[i]
  N += 1

tend = time.perf_counter()

# Check for convergence
if N < maxiters:
  print(f'Jacobi converged')
  print(f'Number of iterations = {N:5d}. || b - A x || = {res:.6g}')
  print(f'Times: {tend-tbeg} seconds')
  xaxis = range(N+1)
  plt.plot(xaxis, resvec[:N+1])
  plt.ylim([0, 125])
  plt.xlabel('Iteration ')
  plt.ylabel(r'$\| b - A x^{(k)}\|_2$')
  plt.show()
else:
  print(f'Jacobi did not converge')
```

Listing A.3: Jacobi method

Python code for Gauss-Seidel method

```python
import scipy.io as sio
import scipy.sparse
import numpy as np
from numpy import linalg as LA
import matplotlib.pyplot as plt
import time

#Define a matrix A
mat = scipy.io.loadmat('bcsstk02.mat')
A = mat['Problem'][0][0][1]
n, nc = A.shape
b = np.ones(n)

# Stop if || b - A x_k || < tol
tol = 1.0e-4

# Allow at most maxiters iterations
maxiters = 4000

# Initial guess is x = 0

x = np.zeros(n)
xnew = np.zeros(n)
resvec = np.zeros(maxiters)
N = 0
tbeg = time.perf_counter()
while N < maxiters:
  res = np.linalg.norm(b - A.dot(x))
  resvec[N] = res
  for i in range(n):
    xnew[i] = b[i]
    for j in range(i)    : xnew[i] -= A[i,j] * xnew[j]
    for j in range(i+1,n): xnew[i] -= A[i,j] * x[j]
    xnew[i] /= A[i,i]

  if LA.norm(b - A @ xnew) < tol:
    break

  for i in range(n): x[i] = xnew[i]
  N += 1
tend = time.perf_counter()

if N < maxiters:
  print(f'Gauss-Seidel converged')
  print(f'Number of iterations = {N:5d}. || b - A x || = {res:.6g}')
  print(f'Times: {tend-tbeg} seconds')

  xaxis = range(N+1)
  plt.plot(xaxis, resvec[:N+1])
  plt.ylim([0, 125])
  plt.xlabel('Iteration ')
  plt.ylabel(r'$\| b - A x^{(k)}\|_2$')
  plt.show()

else:
  print(f'Gauss-Seidel did not converge')
```

Listing A.4: Gauss-Seidel method

Python code for SOR method

```python
import matplotlib.pyplot as plt
import scipy.io as sio
import scipy.sparse
from scipy.sparse.linalg import spsolve
import numpy as np
import time

# Set up a test problem
mat=scipy.io.loadmat('bcsstk01.mat')
A = mat['Problem'][0][0][1]
print(type(A))
n = A.shape[0]
b = np.ones(n)
x0 = np.zeros(n)

# Use the relaxation parameter omega
omega = 1.8

# Iterations will stop when || b - A x_k || < tol
tol = 1.0e-4

# ... or maxit steps have been attempted
maxit = 4000

# Keep track of the residual at each iteration
resvec = np.zeros(maxit)

# The SOR iterations
x = x0[:]
N = 0
tbeg = time.perf_counter()
while N < maxit:
  res = np.linalg.norm(b - A.dot(x))
  resvec[N] = res
  if res < tol: break

  for i in range(n):
    s = 0
    for j in range(i):
      s = s + A[i,j] * x[j]
    for j in range(i+1,n):
      s = s + A[i,j] * x[j]

  x[i] = (omega / A[i,i]) * (b[i] - s) + (1 - omega) * x[i]
  N = N + 1
  tend = time.perf_counter()

# Check for convergence
if N < maxit:
  print(f'SOR converged')
  print(f'Number of iterations = {N:5d}. || b - A x || = {res:.6g}')
  print(f'Times: {tend-tbeg} seconds')
  xaxis = range(N+1)
  plt.plot(xaxis, resvec[:N+1])
  plt.ylim([0, 125])
  plt.xlabel('Iteration ')
  plt.ylabel(r'$\| b - A x^{(k)}\|_2$')
  plt.show()
else:
  print(f'SOR did not converge')
```

Listing A.5: SOR method

Python code for Cholesky factorization

```
import scipy
import scipy.io as sio
import numpy as np
import time
from scipy.sparse.linalg import splu

mat = scipy.io.loadmat('bcsstk16.mat')
A = mat['Problem'][0][0][1]
n, nc = A.shape
b = np.ones(n)

tbeg = time.perf_counter()
B = splu(A)
x=B.solve(b)
tend = time.perf_counter()

print(np.allclose(A @ x - b, np.zeros(n),rtol=1e-04, atol=1e-04))
print(tend-tbeg)
print(np.linalg.norm(b - A.dot(x)))
print(x)
```

Listing A.6: Cholesky factorization

Python code for Conjugate Gradient method

```
import numpy as np
from scipy.sparse.linalg import cg
import scipy
import scipy.io as sio
import time

mat = scipy.io.loadmat('bcsstk16.mat')
A= mat['Problem'][0][0][1]
n, nc = A.shape
b = np.ones(n)

tbeg = time.perf_counter()
x, exit_code = cg(A, b,tol=1e-05)
tend = time.perf_counter()

print(exit_code)
print(np.allclose(A.dot(x)- b, np.zeros(n),rtol=1e-04, atol=1e-04))
print(tend-tbeg)
print(np.linalg.norm(b - A.dot(x)))
print(x)
```

Listing A.7: Conjugate Gradient method

Iterative and Direct methods applied for matrix A

```
import scipy
import time
import numpy as np
from statistics import mean
import scipy.sparse as sparse

fd = open('helmsys.out', 'r');

line = fd.readline()
N, NNZ = (int(x) for x in line.split())

print(N**2, NNZ, 100*NNZ/N**2)

IA = np.zeros(N+1, dtype=int)
for i in range(N+1):
  IA[i] = int(fd.readline()) - 1

JA = np.zeros(NNZ, dtype=int)
for i in range(NNZ):
  JA[i] = int(fd.readline()) - 1

A = np.zeros(NNZ, dtype=np.cdouble)
for i in range(NNZ):
  rp, ip = (float(x) for x in line.split())
  A[i] = complex(rp, ip)

b = np.zeros(N, dtype=np.cdouble)
  for i in range(N):
  rp, ip = (float(x) for x in line.split())
  b[i] = complex(rp, ip)

fd.close()

#Condition number of A
print(np.linalg.cond(A))

#Compute rows' vector
k=-1
Rows = np.zeros(NNZ, dtype = int)
for i in range(NNZ):
if JA[i]<JA[i-1]:
k+=1
Rows[i] = k

#Preconditioner
A = scipy.sparse.csc_matrix((A,Rows,IA))
A_iLU = sparse.linalg.spilu(A)
M = sparse.linalg.LinearOperator((N,N), A_iLU.solve)

Meth_list = ['GMRES', 'BiCGStab', 'SuperLu']
for j in range(len(Meth_list)):
  Times = []
  for i in range(5):
    if j==0:
      tbeg = time.perf_counter()
      x, exitCode = sparse.linalg.gmres(A,b,restart=10,M=M)
      tend = time.perf_counter()
      tdif = tend-tbeg
      Times.append(tdif)
    elif j==1:
      tbeg = time.perf_counter()
      x, exitCode = sparse.linalg.bicgstab(A,b,M=M)
```

```
        tend = time.perf_counter()
        tdif = tend-tbeg
        Times.append(tdif)
      else:
        tbeg = time.perf_counter()
        spilu = scipy.sparse.linalg.spilu(A)
        x = spilu.solve(b)
        tend = time.perf_counter()
        tdif = tend-tbeg
        Times.append(tdif)
    print('Average for ',Meth_list[j], ':', mean(Times))
    print('Converge:', np.allclose(A.dot(x), b))
```