COMPUTATION AND ANALYSIS OF SPECTRA OF LARGE NETWORKS WITH DIRECTED GRAPHS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

AYŞE SARIAYDIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
SCIENTIFIC COMPUTING

JUNE 2010

Approval of the thesis:

COMPUTATION AND ANALYSIS OF SPECTRA OF LARGE NETWORKS WITH DIRECTED GRAPHS

submitted by **AYŞE SARIAYDIN** in partial fulfillment of the requirements for the degree of **Master of Science in Department of Scientific Computing, Middle East Technical University** by,

Prof. Dr. Ersan Akyıldız _____
Director, Graduate School of **Applied Mathematics**

Prof. Dr. Bülent Karasözen _____
Head of Department, **Scientific Computing**

Prof. Dr. Bülent Karasözen _____
Supervisor, **Department of Mathematics, METU**

Prof. Dr. Jürgen Jost _____
Co-supervisor, **Max Planck Institute for Mathematics in the Sciences, Leipzig**

Examining Committee Members:

Prof. Dr. Gerhard Wilhelm Weber _____
Institute of Applied Mathematics, METU

Prof. Dr. Bülent Karasözen _____
Department of Mathematics & Institute of Applied Mathematics, METU

Assist. Prof. Dr. Ömür Uğur _____
Institute of Applied Mathematics, METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: AYŞE SARIAYDIN

Signature :

ABSTRACT

COMPUTATION AND ANALYSIS OF SPECTRA OF LARGE NETWORKS WITH DIRECTED GRAPHS

Sarıaydın, Ayşe

M.S., Department of Scientific Computing

Supervisor : Prof. Dr. Bülent Karasözen

Co-Supervisor : Prof. Dr. Jürgen Jost

June 2010, 88 pages

Analysis of large networks in biology, science, technology and social systems have become very popular recently. These networks are mathematically represented as graphs. The task is then to extract relevant qualitative information about the empirical networks from the analysis of these graphs.

It was found that a graph can be conveniently represented by the spectrum of a suitable difference operator, the normalized graph Laplacian, which underlies diffusions and random walks on graphs. When applied to large networks, this requires computation of the spectrum of large matrices. The normalized Laplacian matrices representing large networks are usually sparse and unstructured.

The thesis consists in a systematic evaluation of the available eigenvalue solvers for nonsymmetric large normalized Laplacian matrices describing directed graphs of empirical networks. The methods include several Krylov subspace algorithms like implicitly restarted Arnoldi method, Krylov-Schur method and Jacobi-Davidson methods which are freely available as standard packages written in MATLAB or SLEPc, in the library written C++.

The normalized graph Laplacian as employed here is normalized such that its spectrum is confined to the range $[0, 2]$. The eigenvalue distribution plays an important role in network analysis. The numerical task is then to determine the whole spectrum with appropriate eigenvalue solvers. A comparison of the existing eigenvalue solvers is done with Paley digraphs with known eigenvalues and for citation networks in sizes 400, 1100 and 4500 by computing the residuals.

Keywords: undirected graph, directed graph, spectral graph theory, eigenvalue, Krylov subspaces, empirical networks

ÖZ

GENİŞ AĞLARDA SİMETRİK OLMAYAN SPEKTRUM HESAPLAMALARI VE ANALİZLERİ

Sarıaydın, Ayşe
Yüksek Lisans, Bilimsel Hesaplama Bölümü
Tez Yöneticisi : Prof. Dr. Bülent Karasözen
Ortak Tez Yöneticisi : Prof. Dr. Jürgen Jost

Haziran 2010, 88 sayfa

Biyoloji, bilim, teknoloji ve sosyal sistemlerdeki geniş ağların analizi son zamanlarda çok popüler olmuştur. Bu ağlar matematiksel olarak çizgiler şeklinde gösterilebilmektedir. Bu tür deneysel ağların çizge şeklinde gösterimleri, ağ yapısıyla ilgili gerekli niteliksel bilgileri çıkarabilmede büyük önem taşımaktadır.

Bir grafik, birimleştirilmiş Laplace gösterimi diye adlandırılan, grafik üzerindeki yayılmaları ve olası gidişleri belirleyen uygun bir fark işleticisinin spektrumu tarafından gösterilebilmektedir. Bu işlem geniş ağlara uygulandığında büyük matrislerin spektrumunun sayısal yöntemlerle belirlenmesi gerekmektedir. Büyük ağları temsil eden birimleştirilmiş Laplace matrisi düzenli bir yapıya sahip olmamaktadır ve doğal ölçeklenme özelliği göstermemektedir.

Bu tezde, deneysel ağların yönlendirilmiş çizgelerini gösteren simetrik olmayan birimleştirilmiş Laplace matrisi için uygun özdeğer yöntemleri karşılaştırılacaktır. Bu metodlar MATLAB ya da C++ daki kitaplık programı olan SLEPc gibi paket programlar olarak serbestçe bulunabilen tam olarak yeniden başlatılmayan Arnoldi metodu, Krylov-Schur metodu ve Jacobi-Davidson metodları gibi çeşitli Krylov alt uzay algoritmalarını kapsamaktadır.

Uygulanan Laplace grafiği normalize edilmiş olup spektrumu $(0, 2)$ aralığından oluşmaktadır. Özdeğer dağılımı ağ analizinde büyük önem taşımaktadır. Sayısal olarak yapılması gereken tüm özdeğerlerin uygun yöntemlerle hesaplanmasıdır. Bu yüzden mevcut yöntemlerin bir karşılaştırması yönlendirilmiş Paley çizgelerinin bilinen özdeğerleriyle ve 400, 1100 ve 4500 büyüklüklerindeki alıntı ağlarıyla kalıntı hesaplanarak yapılacaktır.

Anahtar Kelimeler: yönsüz çizge, yönlendirilmiş çizge, spektral grafik teorisi, özdeğer, Krylov altuzayları, deneysel ağlar

To my family

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Dr. Bülent Karasözen for patiently guiding, motivating, and encouraging me throughout this study.

I am thankful to my co-supervisor Prof. Dr. Jürgen Jost for his hosting in Leipzig and also for his quiding, suggestions and kindness.

Special thanks to Prof. Dr. Ömür Uğur for his helping during installation of programes.

Though I know, it is not possible to express my gratitute to everyone who has a sense during my studies, I would like to thank to a few special people.

My greatest debts are to my family to my admirable father Bahattin, my lovely mother Rukiye, my brother Ahmet and his wife for their love, unfailing support and their patience.

For his endless support and being with me all the time in the days of Leipzig, I am grateful to Murat Sağlam.

I am also grateful to Derya Altıntan and Fatma Yerlikaya for their patiently motivating and listening me.

Also, the environment of Institute of Applied Mathematics (IAM) itself has an important place in this study. Many thanks to Ayşegül İşcanoğlu Çekiç, Sedat Akleylek for their support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Many complex systems in nature have complex network structure [51]. The structure of the complex network displays significant features of complex systems and worth an extensive study. The network structure of complex systems can be treated with graph theory. To analyze these systems structural relation parameters are introduced [30, 32, 73]. Then, they are represented by matrices to exploit the relation between spectrum of these matrices and the structure of complex systems. However, all these procedures can only capture some certain qualitative properties, not all of them. The normalized Laplacian spectrum [5, 8] is a significant tool in this respect. Its spectrum reflects properties of the network structure such that the source of the network can be recognized.

Recent developments about spectral properties of normalized Laplacian matrix require computation of the all eigenvalues of large matrices. Since the large matrices obtained from network applications are sparse and unstructured matrices, direct solvers for eigenvalue problems using finite number of operations are not appropriate because of computational cost and memory limitations. Therefore, iterative solvers are preferred for eigenvalue computation of large sparse matrices as they make use of sparse structure of large matrices. Krylov subspace algorithms and Jacobi-Davidson methods are two widely used iterative methods which make use of sparse structure of matrices.

There have been various softwares applying these methods and most of them are freely available. They may be different in terms of the language (C + +, Fortran or MATLAB), version or simply the design of algorithms. For example, the most popular and widely used sparse matrix eigensolver ARPACK is written in FORTRAN and also it has a C++ interface and implements a famous Krylov subspace method called implicitly restarted Arnoldi method.

In this thesis the comparison of the existing sparse eigenvalue solvers is provided by means of directed graphs. Also, the spectrum of normalized Laplacian matrices of directed graphs describing large networks is computed with the existing eigenvalue solvers since the eigenvalue distribution of this matrix reveals important information about the structural properties of network.

The outline of the thesis is as follows:

Chapter 2 starts with a brief introduction to the study of complex systems with network theory. The development of network modeling is described. Then, basic definitions and elementary notions of undirected and directed graphs needed for the following chapters are given. Also, some necessary tools for spectral analysis of both type of graphs are introduced. Then, general and recent results about relations between spectrum of normalized Laplacian matrix and structure of networks for both types of graphs are given.

In Chapter 3, basic notations and methods for eigenvalue problems are given. The Krylov subspace algorithms with their variations implicitly restarted Arnoldi method, Krylov-Schur methods and augmented block Householder Arnoldi method are introduced. Then Jacobi-Davidson methods are presented with their a new different application Jacobi-Davidson type QR algorithm. A schematic description is done for each type of methods. Applications of methods for symmetric matrices are briefly introduced. The comparison of methods with each other is also provided.

In Chapter 4, the MATLAB implementations of Krylov subspace algorithms: implicitly restarted Arnoldi method (speig), augmented block Householder Arnoldi method (ahbeigs) and for the Jacobi-Davidson type QR algorithm (jdqr) are introduced. Also, the SLEPc library in C++ for the Krylov subspace methods is presented. The parameters used in each package and the usage of the eigenvalue solvers are described.

The thesis concludes with Chapter 5, where the numerical results are given. Firstly, the comparison results for eigenvalue solvers described in Chapter 4 are presented. During this comparison a special digraph called Paley digraph is used in addition to some other network examples. Then, the spectral analysis of citation networks in sizes 400, 1000 and 4500 are done. These networks are analyzed by means of the spectral density plots to investigate the properties of networks using the knowledge described in Chapter 2. Also, some spectral density

plots are provided for a comparison between the spectrum of directed graphs and the underlying undirected graphs. Three dimensional spectral density plots are represented. Finally, the main conclusions for the spectral analysis of directed graphs are described and possible future researches are proposed.

CHAPTER 2

SPECTRAL GRAPH THEORY OF UNDIRECTED AND DIRECTED GRAPHS

Complex systems emerge in many disciplines like bioscience, neuroscience, physics, computer science, artificial life, economics, earthquake prediction, heart cell synchronization, immune systems or reaction-diffusion systems. They have been extensively studied with the help of network theory which is a useful tool for analyzing complex systems. From biological, social, economical, ecological, and technical systems, networks are constructed by means of the components and interactions between the components of the system. Also, they are used to analyze the related data underlying them. The world-wide-web (WWW), scientific collaborations, protein-protein interactions, food webs, nervous systems, gene regulation and metabolic reactions are some examples of complex networks. For various examples and extended research see [6, 7, 51] and [9].

In a complex system, there are many components interacting with each other. The interaction inside the system occurs between close neighbors, however, since the system is coupled, the properties and construction of the system have been affected by each interaction. The interactions may follow some dynamics such that, the system takes some inheritance structure on. So the dynamics influence the structure of the system on one hand or the structure reflects some properties of the dynamics on the other hand. Since the relation between these two properties are very close to each other, it may be useful to analyze the structure and inheritance properties of a complex system [6]. Network theory emerging area of science captures with those properties to some extent.

Complex systems in the form of networks are usually studied by means of graph theory. Graph theory has been playing an important role in analyzing and understanding network structures

since 1735 with the Leonard Euler's solution to the Königsberg bridges problem. In graph theory, the components of a network can be considered to be vertices and the relation between the components can be considered to be edges of the graph. In analyzing network structures with the help of graph theoretical tools, one has to decide first what should be chosen as vertices and which kinds of relations between vertices should be considered describing the edge. However, it is better to think about whether the construction of network from a given system would be helpful to study that system or not, otherwise it would not worth building a network. New methodologies, tools have been introduced to analyze networks intensely. Furthermore, old methods from graph theory have been reconsidered. Thus a new research area of science called network science has been emerged which is based on graph theory.

For long time, scientists have thought that networks consist of sets of components with random connections. Afterwards, they tried to model networks with random connections between the components. By means of social structures, Erdös and Renyi proposed a very simple model [25] to construct a random graph that represents a random network where each node has probability p of being connected by an edge. Interesting properties of this graph model for different values of p are investigated in [26, 27]. Many parameters such as degree distribution, average path length, diameter, betweenness, centrality, transitivity, clustering coefficient, etc have been constructed to analyze the structural properties of networks [10, 26, 51, 71]. The random graph model that Erdös and Renyi proposed has a typical kind of degree distribution. To capture the properties of real networks which have a low average path length and high clustering coefficient, Watts and Strogatz proposed a model [71] that exhibits a "small world phenomenon" by randomizing a fraction of edges connecting nodes in a regular lattice. Networks generated by this model also have similar degree distributions as Erdös and Renyi's random graphs such that they have a more pronounced peak different from real networks as they have a power-law degree distribution. Degree distributions that follow a power law are scale invariant. Because of this, networks that have a power law degree distribution are called scale-free networks [10] although only their degree distributions are scale-free. The first observation of a power law degree distribution in a real network was made by Price in a network of citations of scientific papers in 1965 [53]. By this model it was explained the emergence of the property of a power law degree distribution which was based on the previous work by Simon on wealth distributions in 1950 was explained[59]. After looking at power law degree distributions in the WWW network, Barabasi and Albert proposed a model [10] to generate

a network that shows scale-free degree distribution property. In this network model which is still in progress, a new node is attached to the network not in random way, by establishing connections towards higher-degree nodes. To obtain a new connection with an existing node i with degree n_i depends on the probability $p_i = \frac{n_i}{\sum_j n_j}$. So the network constructed by attaching a new node to already existing nodes by edges with this preference has power law degree distribution. Since then, to capture the structural properties of real networks many models have been introduced [5, 11, 18, 30].

There exists many parameters and tools in graph theory, but many of them can not capture the all features of networks. To obtain the unique and special properties of a network from a special class and to classify the general qualities that are shared by other network structures could be achieved by means of spectral analysis of graphs [5, 16, 30]. In this respect, the eigenvalues and eigenvectors of matrices obtained from network applications play an important role in order to understand structural properties of networks.

In graph theory, connections between edges come in different forms like undirected, directed, unweighted or weighted. Connections that are non-directional or connections that have an implicit direction and connections that are associated with a weight or not. To distinguish these cases we need to define types of graphs. The term undirected graph indicates a graph where the connections are non-directional. For situations in which the connections are directional, the graphs are called directed graph or digraph. The term weighted graph is used for graphs whose edges are associated with a weight and the term unweighted graph for the graphs whose edges are not associated with a weight. Through this study we call undirected graph as graph and directed graph as digraph and also, we use unweighted digraphs and graphs. We first summarize some basic notations and spectral properties of undirected graphs then we provide a wide overview about the directed graphs.

2.1 Undirected Graphs

In this section, we will make use of definitions in Chapter 1 of books [5] and [32] about spectral graph theory.

Definition 2.1 *An **undirected graph** denoted by G is an ordered pair of two sets, a non empty set $V = V(G)$, called vertex set, consisting of objects $v_1, v_2, \ldots, v_N$ that are called vertices or*

sometimes called nodes and another set $E = E(G)$, called edge set, consisting of edges and one edge connects two vertices.

Definition 2.2 *A vertex u is **adjacent** to vertex v if they are joined by an edge. Two adjacent vertices can be called **neighbors** and denoted by $v \sim u$.*

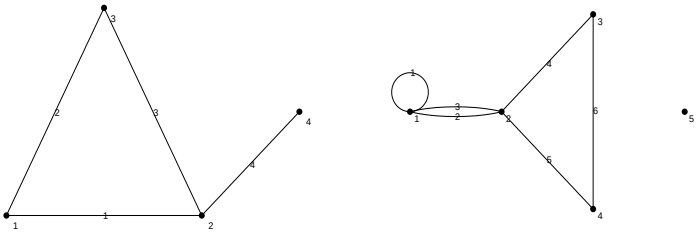
Definition 2.3 *A **multi-edge** is a collection of two or more edges having identical endpoints. The **edge multiplicity** between a pair of vertices u and v is the number of edges between them. A **loop** is an edge that joins a single endpoint to itself.*

Two of the most fundamental notations in graph theory are those of the degree of a vertex and the distance between vertices.

Definition 2.4 *The **degree** of a vertex v in a graph G, denoted by d_v, is the number of edges incident on v plus the number of self-loops.(For simple graphs, of course, the degree is simply the number of neighbors.) The **degree sequence** of a graph is the sequence formed by arranging the vertex degrees into non-decreasing order.*

Definition 2.5 *The **distance** between two vertices in a graph G is the length of the shortest walk between them.*

For example, in the Figure 2.1 (a), the vertex set is $V = \{1, 2, 3, 4\}$ with edges between them. The neighboring vertices are $1 \sim 2$, $1 \sim 3$, $2 \sim 3$, $2 \sim 4$. The edge multiplicity is 1 for all vertices. The degrees with respect to the vertex numbers are 2, 3, 2, 1. In the Figure 2.1 (b), there is a loop, that is, one of the vertices is connected to itself, two vertices have multi-edges and one vertex in isolated as it has no connections.



(a) (b)

Figure 2.1: (a) Simple graph, (b) General graph

Definition 2.6 *A **simple graph** is a graph that has no self-loops or multi-edges. Other graphs in which multi-edges and loops are existing are called **general graphs**.*

There are also many other types of graphs. However, the simple graphs are the most common one as a tool in theoretical graph theory as many problems regarding general graphs can be reduced to problems about simple graphs. In Figure 2.1 (a) and (b) we can see an example of simple graph and general graph respectively.

Definition 2.7 *A **walk** in a graph G is an alternating sequence of vertices and edges*

$$w = v_0, e_1, v_1, e_2, ..., e_N, v_N$$

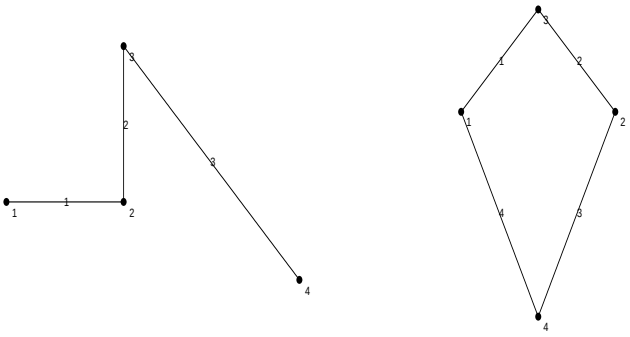
*such that for $j = 1, ..., N$, the vertices v_{j-1} and v_j are the endpoints of the edge e_j. Here, v_0 is the **initial vertex** and v_N is the **terminal vertex**. A vertex which is neither initial nor terminal vertex is called **internal vertex**. A walk is **closed** if the initial vertex is also the final vertex; otherwise, it is **open**. The **length of a walk** is the number edges.*

For example, the walk given in Figure 2.2 (a) can be represented as

$$w = 1, 1, 2, 2, 3, 3, 4$$

where v_j and e_j is represented by numbers here. 1 is the initial vertex and 4 is the final vertex.

Definition 2.8 *A **trail** in a graph G is a walk such that no edge occurs more than once. A **Eulerian trail** in a graph G is a walk that contains each edge of G exactly once. A **path** in a graph is a trail such that no internal vertex is repeated. A **cycle** is a closed path of length at least* 1.



(a) (b)

(c)

Figure 2.2: (a) A walk, (b) A cycle, (c) Trail

In Figure 2.2 (b), we can see an example of cycle whose initial and the final vertices are the same, also Figure 2.2 (c) shows a trail as all edges are distinct.

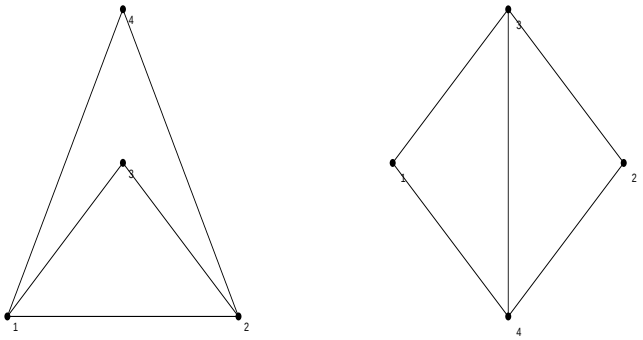
Definition 2.9 *A graph G is **connected** if there is a walk between every pair of vertices.*

Definition 2.10 *The **eccentricity** of a vertex u in a connected graph is its distance to vertex farthest from v. The **radius** of a connected graph is its minimum eccentricity. The **diameter** of a connected graph is its maximum eccentricity.*

Definition 2.11 *A **trivial graph** is a graph consisting of one vertex and no edges.*

Definition 2.12 *Let G and H be two graphs. They are called **isomorphic** if there exists a bijection $\omega : V_G \to V_H$ that has the adjacent vertices relation, i.e., $u \sim v \Leftrightarrow \omega(u) \sim \omega(v)$,*

for all u, v. See Figure 2.3. If there exists an isomorphism from a graph G to itself, then G is called ***symmetric***.



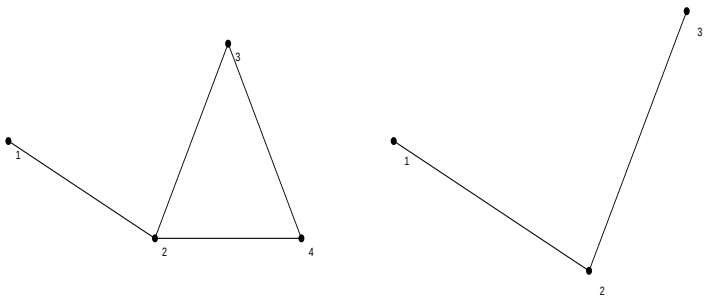
(a) (b)

Figure 2.3: Two isomorphic graphs

Definition 2.13 *A* ***subgraph*** *of a graph G is a graph H such that $V_H \subset V_G$ and $E_H \subset E_G$.*

For example, in Figure 2.4 (b) is a subgraph of (a).

Definition 2.14 *A* ***tree*** *is a connected graph with no cycle.*



(a) (b)

Figure 2.4: (a) Graph (b) A subgraph of (a)

Now, we will give definitions of some important graphs.

Definition 2.15 *A **complete graph** is a simple graph that every pair of vertices are connected with an edge. It is denoted by K_n where n represents the number of vertices.*

Definition 2.16 *A graph G is called a **bipartite graph** if the vertex set $V(G)$ can be decomposed into two disjoint subsets V_1 and V_2, such that each edge of G connects a vertex in V_1 with a vertex in V_2. Hence there is no edge which joins two vertices in the same subset. If every vertex of one set is connected by edges with all vertices of other subset, then the bipartite graph is called a **complete bipartite graph**, and is usually denoted by $K_{m,n}$, where m and n are called the cardinalities of the two subsets.*

Definition 2.17 *A graph is called **d-regular** if each of its vertices have degree d. A regular graph $G = (V, E)$ with degree d is strongly regular if every adjacent vertex has the same number of common neighbors a and every nonadjacent vertex has the same number of common neighbors c. A strongly regular graph is denoted by (n, d, a, c).*

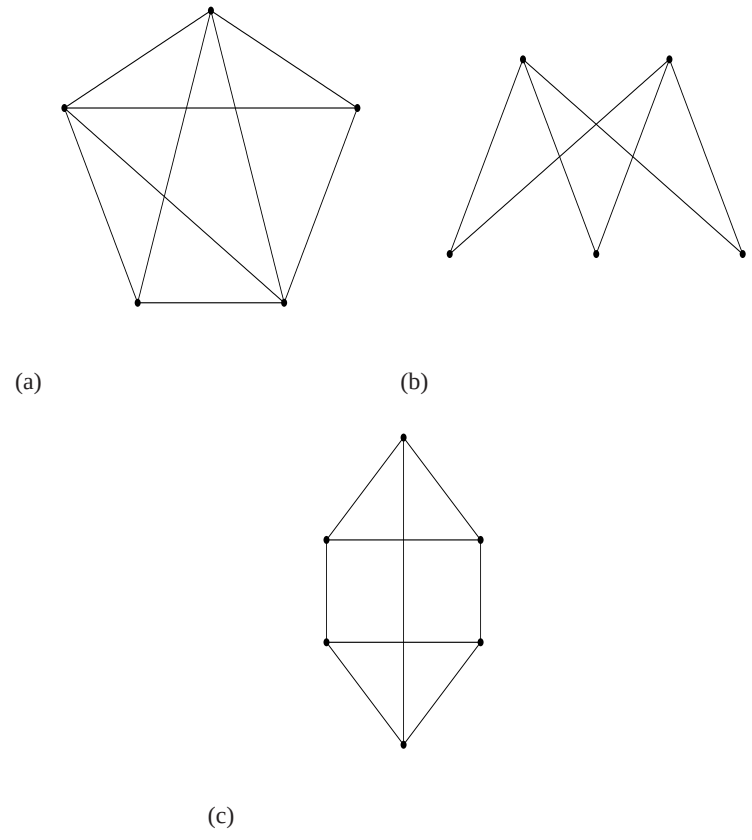


Figure 2.5: (a) Complete graph, (b) Complete bipartite graph $K_{3,2}$ of 5 vertices and (c) Regular graph of degree 3

2.1.1 Connectivity Matrices

A graph $G = (V, E)$ is represented by different kinds of matrices. Eigenvalues of these matrices play an important role in the analysis of graphs. Before describing the spectral properties of graphs, we introduce the connectivity matrices which are the adjacency matrix, Laplacian matrix, and the normalized Laplacian matrix.

Let n_i and n_j be degrees of the vertices i and j of the graph G, respectively. Then connectivity matrices can be defined as follows:

- **Adjacency Matrix:** The matrix A=$[a_{ij}]$ such that

$$a_{ij} = \begin{cases} 1, \text{if ij is an edge,} \\ 0, \text{otherwise,} \end{cases}$$

  is the adjacency matrix of the graph.

- **Laplacian Matrix:** The matrix L=$[l_{ij}]$ such that

$$l_{ij} = \begin{cases} n_i, \text{if i=j,} \\ -1, \text{if ij is an edge,} \\ 0, \text{otherwise,} \end{cases}$$

  is the Laplacian matrix of the graph.

- **Normalized Laplacian Matrix:** There are different kinds of normalized Laplacian matrices with respect to normalization factors. Here, two of them will be introduced. The first one is introduced by Chung [16].

  – The matrix $\mathcal{L} = [l_{ij}]$ such that

$$l_{ij} = \begin{cases} n_i, \text{if i=j and } n_i \neq 0, \\ \frac{-1}{\sqrt{n_i n_j}}, \text{if ij is an edge,} \\ 0, \text{otherwise,} \end{cases}$$

  – The matrix $\Delta = [l_{ij}]$ such that

$$l_{ij} = \begin{cases} 1, \text{if i=j and } n_i \neq 0, \\ \frac{-1}{n_i}, \text{if ij is an edge,} \\ 0, \text{otherwise,} \end{cases}$$

are normalized Laplacian matrix of the graphs.

Unfortunately, there is no clear relationship between the connectivity matrices for most of the graphs. However, the following relations can be used:

$$L = D - A$$

where D is a diagonal matrix with degree on diagonal entries.

$$\mathcal{L} = D^{\frac{-1}{2}} L D^{\frac{-1}{2}} = I - D^{\frac{-1}{2}} A D^{\frac{-1}{2}}$$

and there is similarity between the two kinds of normalized Laplacian matrices:

$$\Delta = D^{\frac{1}{2}} \mathcal{L} D^{\frac{-1}{2}}$$

Example: The connectivity matrices of the graph in Figure 2.6 are as follows:

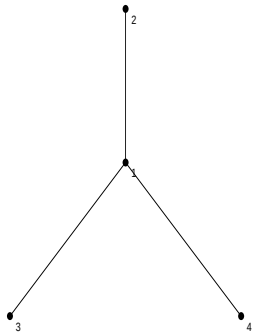


Figure 2.6: A simple graph with 4 vertices

The adjacency matrix:

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

The Laplacian matrix is:

$$L = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}.$$

The normalized matrices are:

$$\mathbf{L} = \begin{pmatrix} 1 & -\frac{1-}{\sqrt{3}} & -\frac{1-}{\sqrt{3}} & -\frac{1-}{\sqrt{3}} \\ -\frac{1}{\sqrt{3}} & 1 & 0 & 0 \\ -\frac{1-}{\sqrt{3}} & 0 & 1 & 0 \\ -\frac{1-}{\sqrt{3}} & 0 & 0 & 1 \end{pmatrix}, \ \Delta = \begin{pmatrix} 1 & -1 & -1 & -1 \\ -\frac{1}{3} & 1 & 0 & 0 \\ -\frac{1}{3} & 0 & 1 & 0 \\ -\frac{1}{3} & 0 & 0 & 1 \end{pmatrix}.$$

2.1.2 Spectral Graph Theory

Spectral graph theory is one of the important tools in spectral analysis of graphs. The spectral analysis of graphs can be done by means of the connectivity matrices. By this way many important properties of a network can be extracted from the eigenvalues of these matrices. The eigenvalue distribution of these matrices shows different information about a network. For instance, the spectrum of adjacency matrix shows details about local structural properties of a graph like number of edges, triangles or loops. The smallest eigenvalues of the Laplacian matrix reveals the connected components in a graph. It can also be used to determine the spanning trees of graphs. The adjacency and Laplacian matrices have played an important role in the early days of spectral graph theory [13, 30, 51]. However, in many practical applications the spectrum of the normalized Laplacian [16, 17] is the most useful one as it shows information about the graph that other connectivity matrices fail to determine. The advantages of the normalized Laplacian are due to the fact that it is consistent with the eigenvalues in the spectral geometry and in stochastic process. By means of normalized Laplacian many results which were only known for regular graphs can be generalized to other graphs. Furthermore, the normalized Laplacian spectrum also provides information about the structural properties of graphs like in protein-protein interaction networks [5].

In this section we will first present some basic facts about the normalized Laplacian operator given in [5, 6, 7, 9, 16].

The normalized Laplacian can be defined as an operator on a connected graph with vertex set

$V = \{i : i = 1 \dots N\}$. Let $u : V \to \mathbb{R}$ be a real-valued function on G such that

$$(u, v) := \sum n_i u(i) v(i).$$

Then the effect of normalized Laplacian of a function defined as above can be reformulated:

$$\Delta u(i) := u(i) - \frac{1}{n_i} \sum_{j, j \sim i} u(j). \tag{2.1}$$

This action of Laplacian on functions reveals three important properties of this operator:

- Δ is a self adjoint operator implying that the operator is symmetric. Therefore, the eigenvalues of normalized Laplacian matrix are real.
- Δ is nonnegative such that all eigenvalues are nonnegative.
- $\Delta u = 0$ when u is constant. This property implies that the smallest eigenvalue is 0.

Since we have assumed that G is connected, the other eigenvalues are greater than 0 i.e., $\lambda_k > 0$ for all $k > 0$. The eigenvalues can be ordered in non-decreasing order such that

$$0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{N-1}.$$

The highest eigenvalue is bounded above with 2, that is, $\lambda_{N-1} \leq 2$ and equality holds if and only if the graph is bipartite. The difference between 2 and the largest eigenvalue λ_{N-1} is an indicator, how different the graph is from a bipartite graph. For a bipartite graph, if λ is an eigenvalue, then $2 - \lambda$ is also an eigenvalue.

If the graph is complete, then the relation

$$\lambda_1 = \lambda_2 = ... = \lambda_{N-1} = \frac{N}{N-1}$$

is satisfied and

$$0 \leq \lambda_1 \leq \frac{N}{N-1} \leq \lambda_{N-1} \leq 2$$

holds. But if the graph is not complete, then $\lambda_1 \leq 1$.

Also, the eigenvalues of a complete bipartite graph $K_{m,n}$ are 0, 1 and 2. The multiplicity of 1 is $m + n - 2$.

If N is the number of vertices in a graph G,

$$\sum_i \lambda_i \leq N,$$

where $i = 0, 1, \ldots, N-1$ and quality holds if and only if the graph is connected.

The number λ_1 is an important eigenvalue of a graph. It shows how difficult it is to cut up the graph into two disjoint components. The Cheeger constant is one of the tools for breaking a graph G into two components G_1 and G_2. This constant was introduced by Cheeger in the context of Riemannian geometry [14]:

$$h(\lambda) = inf\left\{\frac{|E_0|}{min(\sum_{i\in G_1} n_i, \sum_{j\in G_2} n_j)}\right\}$$

where the infimum is taken over subsets E_0 of edges, such that removing E_0 disconnects G into components G_1 and G_2, and $|E_0|$ is the cardinality of E_0.

The relation between λ_1 and the Cheeger constant $h(\lambda)$ for a connected graph is given by

$$\frac{1}{2}h(\lambda)^2 \leq \lambda_1 \leq 2h(\lambda).$$

For a connected graph λ_1 can also be bounded by the diameter and volume such that

$$\lambda_1 \geq \frac{1}{Dvol\,(G)},$$

where D is the diameter of the graph G and $vol\,(G) = \sum_i n_i$.

Up to now the general properties of spectrum of the normalized Laplacian are presented. Recent studies in [5, 7, 8] and [9] show that some graphs have peaks at the eigenvalue 1. It is investigated that the graph evolutionary process is related with these peaks. In the rest of this section these results are summarized. The proofs of the theorems are not provided, however, they can be found in [5].

Combination of the equation (2.1) with the usual eigenvalue equation $\Delta u - \lambda u = 0$ results in

$$\frac{1}{n_i} \sum_{j \sim i} u(j) = (1 - \lambda) u(i), \forall\ i = 1, \ldots, N$$

So if the eigenfunction vanishes at vertices i, the sum of the values of the function on neighbors of i would vanish, $\sum_{j \sim i} u(j) = 0$. On the other hand, when $\lambda = 1$,

$$\sum_{j \sim i} u(j) = 0. \tag{2.2}$$

So, for the eigenvalue 1, there are functions that their sum of values on neighboring vertices are zero. A function u satisfying this property is called balanced solution. The multiplicity of eigenvalue 1 gives the dimension of the set of linearly independent balanced functions on the graph. Furthermore, it can easily be seen from Equation (2.2) that the multiplicity of eigenvalue 1 equals to the dimension of the kernel of the adjacency matrix of the graph.

Definition 2.18 *Let G be a graph. A motif Σ is a connected small subgraph of G containing all edges of G between vertices of Σ.*

Here, the graph G is supposed to be very large when compared with the motif Σ. Following theorems are about the operations on the graph and their effect on spectrum.

Theorem 2.19 *Let G^Σ be obtained from G by adding a copy of the motif Σ consisting of vertices $q_1, \ldots, q_m$ and connections between them and connecting each q_α with all $p \notin \Sigma$ that are neighbors of p_α. Then G^Σ possesses the eigenvalue 1 with a localized eigenfunction that is nonzero only at p_α and q_α.*

Corrollary 2.20 *Let G^Σ be obtained from G by adding a copy of the motif $\Sigma^{'}$, a copy of the motif Σ consisting of vertices $q_1, \ldots, q_m$ and the corresponding connections between them and*

connecting each q_α with all p that are neighbors of p_α. Then G^Σ possesses m more eigenvalues 1 than G with localized eigenfunctions f_1^α ($\alpha = 1, \ldots, m$) that are 1 at p_α, -1 at q_α and zero elsewhere.

This theorem is also satisfied when Σ is a single vertex. Then, according to the corollary if the eigenvalue 1 has a high multiplicity, the graph can be constructed by many vertex doubling or viceversa.

Unfortunately, this result can not be generalized to more general eigenvalues. However, the following theorem has some useful applications:

Theorem 2.21 *Let Σ be a motif in G. Suppose f satisfies*

$$\frac{1}{n_i} \sum_{j \in \Sigma, j \sim i} f(j) = (1 - \lambda) f(i) \tag{2.3}$$

for all $i \in \Sigma$ and some λ. Then the motif doubling of above theorem produces the graph G^Σ with eigenvalue λ and eigenfunction f^{G^Σ} agreeing with f on Σ, with $-f$ the double of Σ and identically 0 on the rest of G^Σ.

This theorem can be applied to the smallest motif of a graph G, an edge. Assume that the vertices of the motif are p_1 and p_2. Then the Equation (2.3) equals to

$$\begin{aligned} \frac{1}{n_{p_1}} f(p_1) &= (1 - \lambda) f(p_2) \\ \frac{1}{n_{p_2}} f(p_2) &= (1 - \lambda) f(p_1) \end{aligned}$$

and admits the solution $\lambda = 1 \pm \frac{1}{\sqrt{n_{p_1} n_{p_2}}}$.

Therefore, as the degree of vertex increases, the eigenvalues gather around 1 more and more, and they are also symmetric with respect to 1. The following theorem is about doubling the entire graph.

Theorem 2.22 *Let G_1 and G_2 be isomorphic graphs with vertices $p_1, \ldots, p_n$ and $q_1, \ldots, q_n$ respectively where p_i corresponds to q_i for all i. Then a graph G_0 can be constructed by connecting p_i with q_j whenever $p_j \sim p_i$. If $\lambda_1, \ldots, \lambda_n$ are eigenvalues of G_1 and G_2 then the new graph has the same eigenvalues as well as the eigenvalue 1 with multiplicity n.*

The next result is about motif joining and works for any eigenvalue.

Theorem 2.23 *Let G_1 and G_2 be graphs with common eigenvalue λ and corresponding eigenfunctions f_λ^1 and f_λ^2. Assume that $f_\lambda^1(p_1) = 0$ and $f_\lambda^2(p_2) = 0$ for some $p_1 \in G_1$ and $p_2 \in G_2$. Then the graph G obtained by joining G_1 and G_2 by identifying p_1 and p_2 also has the same eigenvalue λ with eigenfunction given by f_λ^1 on G_1 and f_λ^2 on G_2.*

2.1.3 Some Special Graphs

In this section, we introduce the eigenvalues of some special graphs and relations in terms of connectivity matrices.

- *Regular Graphs:* There is no explicit formulation for the eigenvalues of regular graphs. However, there is useful formulation between the connectivity matrices of regular graphs:

$$\begin{aligned} L &= dI - A, \\ \mathcal{L} &= I - \frac{1}{k}A, \end{aligned}$$

  where d is the degree of vertices, A is adjacency, L is Laplacian and $\mathcal{L}$ is normalized Laplacian matrices of graphs.

- *Strongly Regular Graphs:* The eigenvalues of the adjacency matrix of strongly regular graph G are determined by the parameters of the graph. For a strongly regular graph G with parameters (n, d, a, c), the eigenvalues of adjacency matrix of G are determined by the following formulas:

$$\begin{aligned} \theta &= \frac{(a-c) + \sqrt{\Delta}}{2}, \\ \tau &= \frac{(a-c) - \sqrt{\Delta}}{2}, \end{aligned}$$

  where $\Delta = (a-c)^2 + 4(k-c)$. The multiplicities of the eigenvalues are

$$m_\theta = \frac{1}{2}\left((n-1) - \frac{2k + (n-1(a-c))}{\sqrt{\Delta}}\right),$$
$$m_\tau = \frac{1}{2}\left((n-1) - \frac{2k + (n-1(a-c))}{\sqrt{\Delta}}\right),$$

- *Paley Graphs:*

Definition 2.24 *Let q be a prime such that* $q \equiv 1$ *(mod 4). The* ***Paley graph*** *[16, 30]* P_q *has a vertex set consisting of elements of GF(q). Two vertices in* P_q *are adjacent if and only if their difference is a square in* $GF(q)$*. Let p be any prime number. The Paley sum graph,* $\widehat{P}_p$*, has vertices* $0, 1, \ldots, p-1$ *and two vertices i and j are adjacent if and only if* $i - j$ *is a quadratic residue module p. For* $p \equiv 3$ *mod(4), Paley graphs are directed such that we introduce in Section 3.5.*

Paley graphs are examples of strongly regular graphs with parameters

$$(q, \frac{q-1}{2}, \frac{q-5}{4}, \frac{q-1}{4}),$$

where q is the number of vertices, $\frac{q-1}{2}$ is the degree of vertices, $\frac{q-5}{4}$ is the number of common neighbors of each vertex and $\frac{q-1}{4}$ is the number of non common neighbors of each vertex. The eigenvalues of the adjacency matrix are

$$\theta_1 = \frac{q-1}{2}, \quad \theta_2 = \frac{-1+\sqrt{q}}{2}, \quad \theta_3 = \frac{-1-\sqrt{q}}{2}$$

with θ_1 of multiplicity 1 and θ_2 and θ_3 have equal multiplicities $\frac{q-1}{2}$.

2.2 Directed Graphs

In this section, basic definitions and properties of digraphs from Chapter 1 of the book [32], Chapter 8 of the books [12] and [29] are given and some spectral properties of them are introduced.

Definition 2.25 *A digraph D consists of two sets, a non empty set $V = V(D)$, called* ***vertex set****, consisting of objects $\{v_1, v_2, ..., v_N\}$ that are called* ***vertices*** *and another $E = E(D)$, called* ***edge set****, consisting of directed edges $\{e_1, e_2, ..., e_N\}$. If $e = (v_i, v_j)$ is an edge of D, then v_i is the* ***initial vertex*** *and v_j is the* ***terminal vertex****. Two directed edges e and e' of a digraph D are said to be* ***parallel*** *if e and e' have the same initial vertex and the same terminal vertex.*

Definition 2.26 *The* ***out degree*** *of a vertex is the number of edges of which it is the initial vertex; the* ***in degree*** *is the number of edges of which it is the terminal vertex. The out degree vector of D is the vector $R = (r_1, r_2, ..., r_N)$ where r_i is the out degree of vertex v_i for $i = 1, 2, ..., N$. The in degree vector is $S = (s_1, s_2, ..., s_N)$ where s_i is the in degree of vertex v_i for $i = 1, 2, ..., N$. Thus R is the row sum vector of the adjacency matrix A of D, and S is the column sum vector. The* ***maximum degree*** *Δ of a digraph D is the maximum integer that occurs among its in degrees and out degrees.*

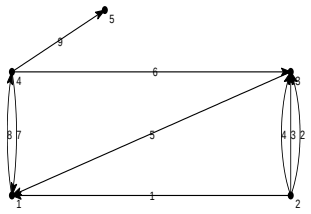
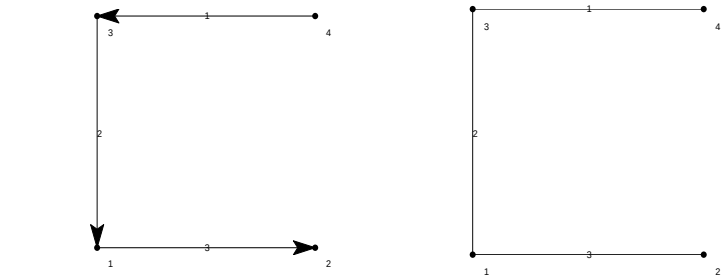


Figure 2.7: Directed graph with 5 vertices and 8 edges

In the Figure 2.7, an example of a directed graph is given with 5 vertices and 9 edges. For the edge $1, 2$ is the initial vertex and 1 is the final vertex. The edges $2, 3, 4$ are parallel as they have the same initial and final vertex. The in degrees and out degrees of each vertex are the followings:

vertex number	in degree	out degree
1	3	1
2	0	4
3	4	1
4	1	2
5	1	0

Definition 2.27 *If D is a digraph, the graph obtained from D by removing the arrows from the directed edges is called the* ***underlying graph*** *of D. This graph is also called the undirected graph corresponding to D.*



(a) (b)

Figure 2.8: (a) A digraph, (b) An underlying graph of (a)

Figure 2.8 (a) shows a digraph and (b) is the underlying graph of it, that is, undirected graph underlying in this digraph.

Definition 2.28 *If v is a vertex of a digraph D, then v is called an* ***isolated vertex*** *of D if* $d_{in}(v) = d_{out}(v) = 0$. *If v is a vertex of a digraph D then v is called a* ***source*** *of D if* $d_{in}(v) = 0$. *If v is a vertex of a digraph D then v is called a* ***sink*** *of D if* $d_{out}(v) = 0$.

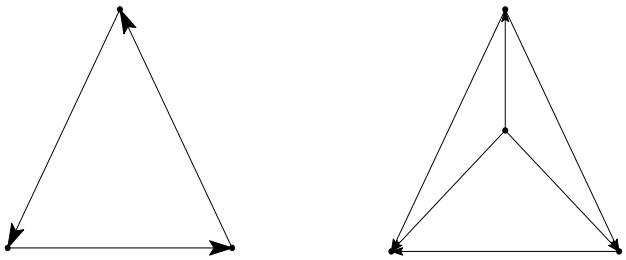
In Figure 2.7, there is no isolated vertex. However, the vertex 2 is an source and the vertex 5 is a sink.

Definition 2.29 *A* ***directed walk*** *in a digraph D is a sequence of the form* $\{v_0, e_1, v_1, e_2 \ldots e_N, v_n\}$ *where* $v_0, v_1, ..., v_n$ *are vertices of D in some order and* $e_1, e_2, ..., e_n$ *are edges of D. A vertex*

can appear more than once in a directed walk but not an edge. An open directed walk in which no vertex is repeated is called a ***directed path****. A closed directed walk in which no vertices, except the initial and final vertices are repeated is called a* ***directed circuit*** *or a* ***directed cycle****. The number of edges present in a directed walk, directed path, directed circuit is called its* ***length****.*

In Figure 2.7, examples of these definitions can easily be seen.

Definition 2.30 *A* ***tournament*** *is an oriented complete graph. Tournaments with two and three vertices are shown in Figure 2.9.*



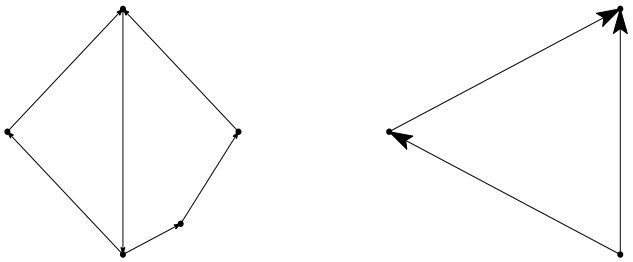
(a) (b)

Figure 2.9: Tournaments with 3 and 4 vertices

Definition 2.31 *A digraph D is said to be* ***strongly connected*** *if there is at least one directed path from every vertex to every other vertex.*

Definition 2.32 *A digraph D is said to be* ***weakly connected*** *if its corresponding undirected graph is connected but D is not strongly connected.*

Figure 2.10 shows an example for strongly connected and weakly connected digraphs.



(a) (b)

Figure 2.10: (a) Strongly connected (b) Weakly connected digraph

Definition 2.33 *A digraph that has no self-loop or parallel edges is called a* ***simple digraph****.*

Definition 2.34 *Digraphs that have at most one directed edge between a pair of vertices, but are allowed to have self-loops, are called* ***asymmetric*** *or* ***antisymmetric*** *digraph.*

Definition 2.35 *Digraphs in which for every edge* (u, v) *(i.e., from vertex u tov) there is also an edge* (v, u)*.*

Definition 2.36 *Isomorphic graphs are defined such that they have identical behavior in terms of graph properties. In other words, if their labels are removed, two isomorphic graphs are indistinguishable. For two digraphs to be isomorphic not only their corresponding undirected graphs must be isomorphic, but the directions of the corresponding edges must also agree. Two digraphs* D_1 *and* D_2 *are said to be* ***isomorphic*** *if both of the following conditions hold:*

- *The underlying graphs of* D_1 *and* D_2 *are either identical or isomorphic.*
- *Under the one-to-one correspondence between the edges of* D_1 *and* D_2 *the directions of the corresponding edges are preserved.*

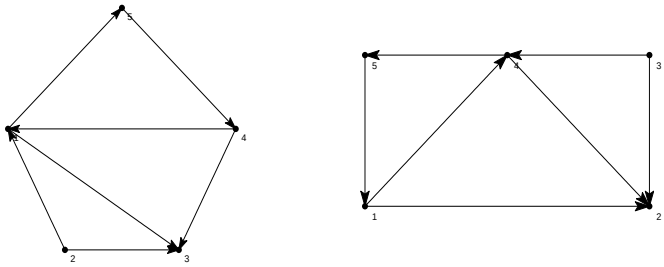
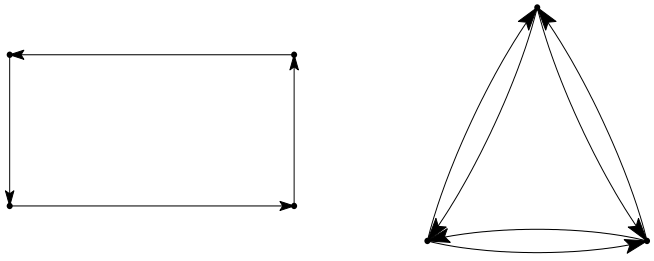


Figure 2.11: Two isomorphic digraphs

Definition 2.37 *A digraph D is said to be a* ***balanced digraph*** *or an* ***isograph*** *if* $d_{in}(v) = d_{out}(v)$ *for every vertex v of D.*

Definition 2.38 *A balanced digraph is said to be* ***regular*** *if every vertex has the same in-degree and out-degree as every other vertex.*



(a) (b)

Figure 2.12: (a) 1-regular digraph (b) Complete digraph

Definition 2.39 *A complete undirected graph was defined as a simple graph in which every vertex is joined to every other vertex exactly by one edge. A* ***complete digraph*** *is a simple digraph such that between each pair of its vertices oppositely directed edges exist.*

2.2.1 Connectivity Matrices

Digraphs (directed graphs) are also represented by using adjacency matrix, Laplacian matrix or the normalized Laplacian matrix similar to graphs (undirected graphs). However, the definitions of these matrices are different from those for graphs and spectral analysis of digraphs

with these matrices is a very active research area in graph theory [12, 19, 20, 50]. Here, the definitions of these matrices are given before going on to the spectral analysis of digraphs.

- **Adjacency Matrix:** The matrix A=[a_{ij}] such that

$$a_{ij} = \begin{cases} 1, \text{if there is an edge from i to j,} \\ 0, \text{otherwise.} \end{cases}$$

  is the adjacency matrix of the digraph.

  Unfortunately, there is no explicit definition for the Laplacian and the normalized Laplacian matrices of digraphs which is similar to definitions for undirected graphs. Instead, the following relation is used.

- **Laplacian Matrix:** The matrix L=[l_{ij}] such that

$$L = D - A$$

  is the Laplacian matrix of the graph where A is the adjacency matrix of digraph and D is the diagonal matrix with in degrees or out degrees on diagonal entries.

  The Laplacian matrix of a digraph can also be defined by using the transition probability matrix. This definition is done by Chung in [20].

  For a given digraph D, a typical transition probability matrix $P = P_D$ is defined as

$$P_{(i,j)} = \begin{cases} \frac{1}{n_i}, \text{ if ij is an edge,} \\ 0, \text{otherwise} \end{cases}$$

  When digraph is weighted, that is, its each edge has weights $w_{ij} \geq 0$, a general transition probability matrix P can be defined as

$$P_{(i,j)} = \frac{w_{ij}}{\sum_k w_{ik}},$$

  where $k = 1, \ldots, N$.

  An unweighted digraph is just a special case with weights having the values 1 or 0.

  An irreducible matrix M with non-negative entries has a unique (left) eigenvector with positive entries according to Perron-Frobenius theorem. Assume that ρ denote the

eigenvalues of the positive eigenvector of P. Then, the absolute values of all eigenvalues of M are bounded above ρ. This situation can be extended to digraphs. The transition probability matrix P of a strongly connected digraph has a unique left eigenvector ϕ used as a row vector with $\phi(v) > 0$ for all edges v, and

$$\phi P = \rho\phi.$$

Since the 1-vector satisfies the relation $P1 = 1$, we have $\rho = 1$ and according to the Perron-Frobenius theorem all other eigenvalues of P have absolute value at most 1. We can normalize and choose ϕ satisfying

$$\sum_{v} \phi(v) = 1.$$

Here, ϕ is called the Perron vector of P. For a general digraph, there is no closed form solution of ϕ. So, the Laplacian matrix of a digraph D is defined as

$$L = I - \frac{\Phi^{1/2} P \Phi^{-1/2} + \Phi^{-1/2} P^* \Phi^{1/2}}{2},$$

where ϕ is a diagonal matrix with entries $\Phi(v, v) = \phi(v)$ and P^* denotes the conjugated transpose of P. Clearly, it satisfies the relation

$$L^* = L.$$

- **Normalized Laplacian Matrix:** For any digraph D associated with an adjacency matrix A, the normalized Laplacian matrix $\mathcal{L} = [l_{ij}]$ is defined as

$$\mathcal{L} = I - D^{-1}A.$$

Here, $\mathcal{L}$ has a number of important properties. $D^{-1}A$ is a stochastic matrix i.e., its row sums are all 1 and its possibly complex eigenvalues have modulus in the interval $[0, 1]$. Consequently, the eigenvalues of the normalized Laplacian matrix $\mathcal{L}$ have modulus in the interval $[0, 2]$.

Example: The connectivity matrices of the graph in Figure 2.13 are as follows:

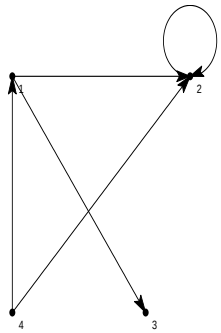


Figure 2.13: A digraph with 4 vertices

The adjacency matrix is

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The Diagonal matrix is

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix},$$

where in degree is used.

The Laplacian matrix is

$$L = \begin{pmatrix} 1 & -1 & 0 & -1 \\ 0 & 2 & 0 & 0 \\ -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

The normalized Laplacian matrix is given by:

$$\mathcal{L} = \begin{pmatrix} 1 & -1 & 0 & -1 \\ 0 & \frac{2}{3} & 0 & 0 \\ -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

2.2.2 Spectral Properties of Digraphs

Spectral analysis of digraphs is also done through the eigenvalues of connectivity matrices. However, it is not well developed as for the graphs. In spectral analysis of digraphs the relation between digraph parameters and eigenvalues of adjacency matrix have been analyzed [12]. In recent years, eigenvalues of the Laplacian matrix have also gained popularity [19, 20]. The normalized Laplacian eigenvalues are also used. However, it is really new for digraphs. In contrast to graphs, the applications for real networks are very few.

In this section, recent studies about the spectral properties of the adjacency matrix and Laplacian matrix are introduced. The proofs of the theorems provided here are not included but they can be found in [12, 19, 20]. Also, the relationship between spectral properties and eigenvalue distribution of digraphs and their underlying graphs are provided [50].

The eigenvalues of the adjacency matrix A are the eigenvalues of digraph D. Since A is not necessarily a symmetric matrix, the eigenvalues of D are, in general, complex numbers $\lambda_1, \lambda_2, ..., \lambda_N$, where we usually assume that

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_N|.$$

The spectral radius of digraph D denoted by $\rho(D)$ is equal to $|\lambda_1|$, the largest absolute value of an eigenvalue of the adjacency matrix A. When digraph D is regular, in this case the spectral radius of D is equal to the in degree or similarly the out degree of regular digraph.

The adjacency matrix A of a digraph D is a nonnegative matrix. Thus, the Perron-Frobenius theory of nonnegative matrices provides information on the spectrum of digraph. Theorem 1.1 in [12] provides some important information. By the following theorem a classical result of the adjacency matrix eigenvalues of digraph is obtained. Remember that the period or index

of imprimitivity of a digraph D is the greatest common divisor d of the lengths of the cycles of D and a cycle of length k of a digraph is a sequence $v_1, v_2, ..., v_N, v_1$ of vertices such that $v_1, v_2, ..., v_N$ are distinct, and $(v_1, v_2), ..., (v_{N-1}, v_N), (v_N, v_1)$ are edges.

Theorem 2.40 *Let D be a strongly connected digraph of order N, then:*

- *The spectrum of D, as a set of points in the complex plane, is invariant under a rotation about the origin by the angle $2\pi/d$*

- *The spectral radius of D satisfies*

$$\min\{r_1, r_2, ..., r_N\} \leq \rho(D) \leq \max\{r_1, r_2, ..., r_N\}.$$

  Also, $\rho(D) = \min\{r_1, r_2, ..., r_N\}$ if and only if $\rho(D) = \max\{r_1, r_2, ..., r_N\}$, and these inequalities hold if and only if digraph D has a constant out degree vector such that $R = \{r_1, r_2, ..., r_N\}$. A similar conclusion holds using the in degree vector S in place of the out degree vector R.

- *If D' is a digraph obtained from D by deleting one or more edges, then $\rho(D') < \rho(D)$.*

It is known that a digraph D is bipartite when its vertex set can be partitioned into two sets V_1 and V_2 such that each edge has its initial vertex in V_1 and its terminal in V_2. Thus, the adjacency matrix of a bipartite digraph is of the form

$$A = \begin{pmatrix} 0 & A_1 \\ A_2 & 0 \end{pmatrix},$$

where the zero matrices are square matrices of order $|V_1|$ and $|V_2|$, respectively. We have

$$A^2 = \begin{pmatrix} A_1A_2 & 0 \\ 0 & A_1A_2 \end{pmatrix},$$

where A_1A_2 have the same nonzero eigenvalues. The Perron-Frobenius theory of nonnegative matrices generalizes a result for bipartite graphs in [12] by the following theorem.

Theorem 2.41 *[12] A digraph is bipartite if and only if its spectrum is invariant under multiplication by* -1*, equivalently, if and only if the spectral radius of D is the nonnegative of some eigenvalue D.*

Recent studies in [19, 20] show that eigenvalues of the Laplacian matrix provide some information about the spectral properties of digraphs such as Cheeger constant, Cheeger inequality or diameter. In the rest of this section these spectral properties are introduced.

For a directed graph D with the transition probability matrix P and the Perron vector ϕ, which are given above, the Rayleigh quotient for any $f : V(D) \to \mathbb{C}$ is defined as

$$R(f) = \frac{\sum_{u\to v} |f(u) - f(v)|^2 \phi(u) P(u,v)}{\sum_v |f(v)|^2 \phi(v)}$$

The Laplacian matrix of a digraph D is

$$L = I - \frac{\Phi^{1/2} P \Phi^{-1/2} + \Phi^{-1/2} P^* \Phi^{1/2}}{2},$$

where Φ is a diagonal matrix with entries $\Phi(v,v) = \phi(v)$ and P^* is the conjugated transpose of P.

Theorem 2.42 *[20] For a directed graph D with the transition probability matrix P suppose the Rayleigh quotient and the Laplacian are defined as above. Then we have*

$$\begin{aligned} R(f) &= 2\frac{\langle fL, f\rangle}{\langle f\Phi, f\rangle} \\ &= 2 - \frac{f(\Phi P + P^*\phi)f^*}{f\Phi f^*} \\ &= 2\frac{\langle gL, g\rangle}{\|g\|^2} \end{aligned}$$

where $g = f\phi^{1/2}$.

A consequence of this theorem is the following:

Corollary 2.43 *[20] Suppose a directed graph D has the Laplacian eigenvalues* $0 = \lambda_0 \leq \lambda_1 \leq, ..., \leq \lambda_{N-1}$*. Then the eigenvalues and the Rayleigh quotient are related as follows:*

- *For λ_1, we have*

$$\begin{aligned}\lambda_1 &= \inf_{\sum_x f(x)\phi(x)=0} \frac{R(f)}{2} \\ &= \inf_f \sup_c \frac{\sum_{u\to} v|f(u)-f(v)|^2\phi(u)P(u,v)}{2\sum_u |f(v)-c|^2\phi(v)},\end{aligned}$$

  where ϕ is the Perron vector and Φ is the Perron diagonal matrix.

- *Suppose ϕ_i is an eigenvector of the Laplacian associated with eigenvalue λ_i. For $f_i = \phi_i\Phi^{-1/2}$, we have*

$$\begin{aligned}\lambda_i f_i(x)\phi(x) &= \left[f_i(\Phi - \frac{\Phi P + P^*\Phi}{2})\right](x) \\ &= \phi(x)f_i(x) - \frac{\sum_{y\to x} f_i(y)\phi(y)P(y,x)}{2} - \frac{\sum_{x\to y} f_i(y)P(x,y)\phi(x)}{2} \\ &= \frac{1}{2}\sum_y (f_i(x)-f_i(y))(\phi(y)P(y,x)+P(x,y)\phi(x))\end{aligned}$$

  for each vector x.

Theorem 2.44 *[20] For a directed graph D, the eigenvalue λ_1 of the Laplacian L is related to the eigenvalues ρ_i of the transition probability matrix P as follows.*

$$\lambda_1 \le \min_{i\neq 0}(1 - Re(\rho_i))$$

where $i = 1,\ldots,N$ and $Re(x)$ denotes the real part of the complex number x.

It is also possible to define the Cheeger constant and the Cheeger inequality for digraphs by means of Laplacian matrix. Let S denote a subset of vertices of the directed graph D. The *out-boundary* of S, denoted by ∂S, consists of all edges (u,v) with $u \in S$ and $v \notin S$

$$F(\partial S) = \sum_{u\in S, v\notin S} F(u,v).$$

If F is a circulation, it satisfies

$$F(\partial S) = F(\partial \bar{S}),$$

where $\bar{S}$ denotes the complement of S.

For a vertex v, we define $F(v) = \sum_{u,u\to v} F(u,v)$ and $F(S) = \sum_{v\in S} F(v)$. So, for a strongly connected digraph D with stationary distribution ϕ, if we consider the circulation flow F_ϕ, the Cheeger constant as in the following form

$$h(D) = \inf_S \frac{F_\phi(\partial S)}{\min\left\{F_\phi(S), F_\phi(\bar{S})\right\}}$$

where S ranges over all non-empty proper subset of the vertex set of D. Here, h can be related to the eigenvalues of the Laplacian by establishing the directed analog of the Cheeger inequality. The following theorem explains this situation.

Theorem 2.45 *[20] For a directed graph D with eigenvalues λ_i of the Laplacian. Then $\lambda = \min_{i\neq 0} |\lambda_i|$ satisfies*

$$2h(D) \geq \lambda \geq \frac{h^2(G)}{2},$$

where $h(D)$ is the Cheeger constant of D.

Following theorems give lower bounds for the Cheeger constants for various families of digraphs is provided:

Theorem 2.46 *[20] For a strongly connected regular directed graph D on n vertices and degree k, we have*

$$h(D) \geq \frac{2}{kn}.$$

Theorem 2.47 *[20] For a strongly connected Eulerian directed graph D on m edges, we have*

$$h(D) \geq \frac{2}{m}.$$

Theorem 2.48 *[20] For some directed graphs D with bounded out-degrees, the Cheeger constant of D can be exponentially small, i.e.,*

$$h(D) \leq c^{-n}$$

for some constant c.

It has been known that the diameter of graphs can be bounded using the eigenvalues of the Laplacian matrix. In this situation a natural question is to see if it is feasible to extend these relations to digraphs. In a digraph, the diameter can be naturally defined. It has been known that a directed graph is strongly connected if for any two vertices v_1 and v_2, there is a directed path from v_1 to v_2. The diameter of a strongly connected digraph is defined as the maximum distance among pairs of vertices. If a digraph is not strongly connected then its diameter is taken to be infinity.

Theorem 2.49 *[20] For a strongly connected directed graph D, the diameter diam(D) of D satisfies*

$$diam(D) \leq \left\lfloor \frac{2min_x \log(1/\phi(x))}{log\frac{2}{2-\lambda}} \right\rfloor + 1$$

where λ is the first non-trivial eigenvalue of the Laplacian and ϕ is the Perron vector of the random walk on D.

It is naturally seen that the eigenvalues of the Laplacian is quite useful for capturing various properties which can not done through adjacency matrix eigenvalues of digraphs.

Up to now, the general properties of the spectrum of the adjacency matrix and the Laplacian matrix of a digraph are introduced, which is not simple as the case for graphs. In spectral analysis of graphs, relation between some exact eigenvalues of graphs and their spectral properties is provided. However, there is no such a relationship for digraphs. Instead of this, there is a relationship between some spectral properties of digraphs and the eigenvalue distribution. Furthermore, digraphs and graphs are related according to eigenvalue distribution. From now on, we describe some of these properties.

The Flow in the Network: The connectivity matrices of a digraph are generally nonsymmetric. Therefore, the corresponding eigenvalues are in complex form. A ring structure in the complex spectrum represents the existence of a loop-like flow in the network. This situation can easily be explained by considering the simplest network with a loop-like flow. The eigenvalues of this simplest network can be computed analytically by using the Fourier transform.

Assume that digraph D has n nodes and each node is connected to its k th neighbors at farthest and the edges are ordered in one direction. The eigenvalues of the digraph D can be written in the form:

$$\lambda_n = \sum_{m=1}^{k} e^{i\frac{2\pi n}{N}m}.$$

In the case of $k = 1$, which is the simple one-dimensional directed chain with a periodic boundary condition, we obtain

$$\lambda_n = e^{i\frac{2\pi n}{N}m}.$$

This constitutes the unit circle in the complex plane which is the simplest ring structure we can consider.

The one-dimensional directed chain is a directed loop of N steps. If we have a directed loop of N steps, the eigenvalues satisfy

$$\lambda^n = 1.$$

They form a ring structure in the complex plane. When there are more complicated loops, which means regular directed rings with $k \geq 2$, the ring structure is still dominant.

Relation With Graph Spectra: The real and imaginary parts of the eigenvalues of complex eigenvalues of a digraph D have their own information. They also have different roles in graph in that spectral graph theory. If the directions of all edges of a digraph D, the spectral density of the resulting undirected graph is the same as the density of the real part of the eigenvalues of digraph. Thus, the distribution of real part of eigenvalues of a digraph reflects the undirected graph topology.

The graph spectrum of a undirected graph is directly related to the number of loops in the network. Thus, it is known that the variance of the spectral density corresponds to the total number of edges in the network. In directed graphs, there is also a relation between the spectrum and the number of loops. For a digraph D, the variance of the distribution of the

imaginary part of the eigenvalues of digraph approximately gives the number directed edges. For different examples and details see [50].

2.2.3 Some Special Digraphs

In this section, we list some special digraphs and eigenvalues.

- *Regular Digraphs:* There is no explicit formulation for the eigenvalues of regular digraphs as in the case of regular graphs. However, the useful formulation between the connectivity matrices of regular graphs can also be used for regular digraphs:

$$\begin{aligned} L &= dI - A, \\ \mathcal{L} &= I - \frac{1}{k}A, \end{aligned}$$

  where d is the degree of vertices, A is adjacency, L is Laplacian and $\mathcal{L}$ is normalized Laplacian matrices of graphs.

  These equations can be adapted to the eigenvalues of a digraph. Let $\lambda_1, \ldots, \lambda_N$ be eigenvalues of the adjacency matrix of digraph D. Then we have

$$\begin{aligned} \text{spectrum}(L) &= \{d - \lambda_1, \ldots, d - \lambda_N\}, \\ \text{spectrum}(\mathcal{L}) &= \left\{1 - \frac{\lambda_1}{d}, \ldots, 1 - \frac{\lambda_N}{d}\right\}. \end{aligned} \tag{2.4}$$

  Therefore, for a d-regular graph it is enough to determine eigenvalues of one of connectivity matrices.

- *Strongly Regular Digraphs:* The eigenvalues of the adjacency matrix of strongly regular digraphs are determined by digraph parameters. Let D be a digraph with parameters (n, k, t, a, c), where n is the number of nodes, k is the in degree of each vertex, t is the out degree of each vertex and a and c represents the common and uncommon neighbors of each node, respectively.For a strongly regular digraph, as each vertex has the same in degree and out degree we call $k = t = d$. Then the digraph parameters are defined as (n, d, a, c) and the eigenvalues of adjacency matrix of strongly regular digraph D are determined by the following formulas:

$$\theta = \frac{(a-c)+\sqrt{\Delta}}{2},$$
$$\tau = \frac{(a-c)-\sqrt{\Delta}}{2},$$

where $\Delta = (a-c)^2 + 4(d-c)$.

- *Paley Digraphs:*

Definition 2.50 *Let q be a prime such that $q \equiv 3$ (mod 4). The **Paley digraph** [16, 30] P_q has a vertex set consisting of elements of GF(q). Two vertices in P_q are adjacent if and only if their difference is a square in $GF(q)$.*

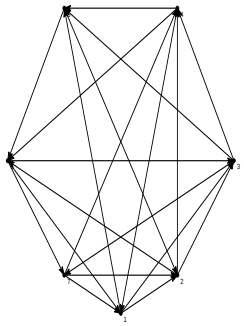


Figure 2.14: (a) Paley digraph with 7 vertices

Paley graphs are examples of strongly regular digraphs. The eigenvalues of the adjacency matrix are

$$\lambda_1 = \frac{q-1}{2}, \quad \lambda_2 = \frac{-1-i\sqrt{q}}{2}, \quad \lambda_3 = \frac{-1+i\sqrt{q}}{2}.$$

with λ_1 of multiplicity 1 and λ_2 and λ_3 have equal multiplicities $\frac{q-1}{2}$ where q is the number of vertices.

Moreover with the relation between matrices of regular digraphs the eigenvalues of Laplacian matrices of Paley digraphs can also be formulated as:

$$\lambda_1 = 0, \quad \lambda_2 = 1 - \frac{1+q}{2(1-i\sqrt{q})} \text{ and } \lambda_3 = 1 + \frac{-1+q}{2(-1+i\sqrt{q})}$$

with the same multiplicities as above.

CHAPTER 3

EIGENVALUE COMPUTATION OF LARGE NONSYMMETRIC MATRICES

Complex networks are represented by graphs or digraphs discussed in Chapter 2 and matrices obtained from network applications do not have a regular structure. They are generally in sparse form, that is, most of the entries of matrices are zero. Different from dense matrices only the nonzero entries are stored. This property provides many advantages when numerical methods are applied to determine eigenvalues and eigenvectors of large matrices.

Most of the numerical methods for eigenvalue computation of large matrices are based on symmetric matrices [21, 31, 52, 74]. On the other hand, the ability to compute eigenvalues and corresponding eigenvectors of large sparse non symmetric matrices is a very active and challenging research area. It is becoming increasingly important in a wide variety of applications. This increasing demand has improved interest in the development of new methods and softwares for numerical solution of eigenvalues and corresponding eigenvectors of large sparse non symmetric matrices. The existence of these new methods and softwares has unable the solution of problems that would not have been posed five or ten years ago. Until very recently, there were nearly no softwares for large-sparse non symmetric problems. Fortunately, this situation is improving rapidly.

In the light of these developments in methods and softwares for large sparse non symmetric problems, in this chapter we provide a wide overview of numerical solution of eigenvalues and corresponding eigenvectors of large sparse non symmetric matrices [23, 55, 56, 63]. The focus will be on two classes of methods called Krylov subspace (projection) methods and Jacobi-Davidson methods [56, 65, 66, 67, 68]. The discussion begins with a brief theory of eigenvalues and basic iterations suitable for large scale problems to motivate the introduction

of Krylov subspaces.Then, Krylov subspace projection methods, implicitly restarted Arnoldi method, Krylov-Schur method and augmented block Householder Arnoldi method, and finally Jacobi-Davidson methods are presented in detail.

3.1 Eigenvalues and Single Vector Iterations

To understand the behavior and limitations of the algorithms, a brief discussion of the mathematical structure of the eigenvalue problem is necessary. In this discussion, the real and complex number fields are denoted by $\mathbb{R}$ and $\mathbb{C}$ respectively. The standard n-dimensional real and complex vectors are denoted by $\mathbb{R}^n$ and $\mathbb{C}^n$. The symbols $\mathbb{R}^{m\times n}$ and $\mathbb{C}^{m\times n}$ denote the real and complex vectors with m rows and n columns. The transpose of a matrix A is denoted by A^T, A^* denotes the complex conjugate, and the symbol $\|.\|$ denotes the Euclidean or 2-norm of a vector.

The elements of the discrete set $\sigma(A) \equiv \{\lambda \in \mathbb{C} : \text{rank}(A - \lambda I) < n\}$ are the eigenvalues of A and they may be characterized as the n roots of the characteristic polynomial $p_A = \det(A-\lambda I)$. There is at least one nonzero vector x such that $Ax = \lambda x$ corresponding to each distinct eigenvalue $\lambda \in \sigma(A)$. This vector is called a right eigenvector of A corresponding to eigenvalue λ. The pair (x, λ) is an eigenpair. A nonzero vector y such that $y^*A = \lambda y^*$ is called a left eigenvector. The algebraic multiplicity $n_a(\lambda)$ of an eigenvalue is of λ is its multiplicity as a root of the characteristic polynomial. The geometric multiplicity $n_g(\lambda)$ of an eigenvalue λ is the number of linearly independent eigenvectors to that eigenvalue. A matrix is defective if $n_g(\lambda) < n_a(\lambda)$ and otherwise it is called non defective. An eigenvalue of algebraic multiplicity 1 is said to be simple.

A subspace S of $\mathbb{C}^{n\times n}$ is called invariant subspace of A if $AS \subset S$. It is straightforward to show if $A \in \mathbb{C}^{n\times n}$, $X \in \mathbb{C}^{n\times k}$, and $B \in \mathbb{C}^{k\times k}$ satisfy

$$AX = XB \tag{3.1}$$

then $S \equiv \text{Range}X$ is an invariant subspace of A. Moreover, if X has full column rank k then the columns of X form a basis for this subspace and $\sigma(B) \subset \sigma(A)$. If $k = n$, then $\sigma(B) = \sigma(A)$, and A is said to be similar to B under the similarity transformation X. A is diagonalizable if it

is similar to a diagonal matrix and this property is equivalent to A being non-defective.

Schur decomposition is a fundamental theorem to discuss the numerical algorithms for eigen-problems.

Definition 3.1 *Every square matrix A possesses a Schur decomposition*

$$AQ = QR,$$

where Q is unitary $Q^T Q = I$ and R is upper triangular. The diagonal elements of R are the eigenvalues of A. The diagonal elements of R_k are the eigenvalues of A.

If V_k represents the leading k columns of Q, and R_k the leading principal $k \times k$ sub matrix of R, then

$$AV_k = V_k R_k.$$

This is called a partial Schur decomposition of A, and there is always a partial Schur decomposition of A with the diagonal elements of R_k consisting of any specified subset of k eigenvalues of A.

3.1.1 Single-vector Iterations

Single-vector iterations are the simplest and most storage-efficient ways of computing a single eigenvalue and corresponding eigenvector. The power method is the oldest and simplest of these methods and also underlies the behavior of all methods for large-scale problems. It only requires multiplication of an arbitrary nonzero vector repeatedly by a matrix A. It converges to the dominant eigenvalue of the matrix A as long as the starting vector has a component in the direction of dominant eigenvector.

The power method seems useful in practice. However, it has two important drawbacks. The convergence rate of the power method proportional to $\frac{\lambda_2}{\lambda_1}$ where λ_2 is the eigenvalue having

second-largest magnitude may be very slow, or may not happen at all and only one eigenvalue and corresponding eigenvector can be computed.

The problem of slow convergence and convergence to interior eigenvalues may be prevented by replacing A by $(A - \sigma I)^{-1}$, where σ is near the eigenvalue of interest. More can be learnt about such spectral transformations from [23, 33, 65, 66, 67]. To address the problem of obtaining several eigenvalues and corresponding eigenvectors, deflation schemes have been used [55]. However, this scheme is not suitable for non symmetric problems. In stead of this scheme , various linear combinations of power iterations can be used to approximate additional eigenvalues and eigenvectors such that there is a systematic way to consider all such possibilities at once and pick the most suitable one automatically.

3.2 Krylov Projection Methods

This section is based on Chapter 4 of book [23] and [55, 63] and articles [62, 64, 70].

The successive vectors produced by a power iteration may contain considerable information along eigenvector directions corresponding to eigenvalues near the one with largest magnitude. A single vector power iteration ignores this useful information. Subspace projection provides a way to extract this additional information. Rather than discard the vectors produced during the power iteration, additional eigen information is obtained by looking at various linear combinations of the power sequence. This leads to consideration of the Krylov subspace

$$K_m(A, x) = \text{span}\left\{x, Ax, A^2x, \ldots, A^{m-1}x\right\}$$

and to look for the best approximate eigenvector that can be constructed from this space. Methods which use linear combinations of vectors in this space are called Krylov subspace or projection methods [23].

The basic idea in Krylov projection methods is to construct approximate eigenvectors in the Krylov subspace $K_m(A, x)$. We define a Ritz pair as any pair (x_i, λ_i) that satisfies the Galerkin condition

$$v^T(Ax_i - \lambda_i x_i) = 0 \text{ for all } \quad v \in K_k(A, v_0).$$

That is, the Ritz pair satisfies the eigenvalue-eigenvector relationship in the projection onto a smaller space. To obtain a good approximation for an eigenpair of A, the component orthogonal to the space must be sufficiently small.

3.2.1 Implicitly Restarted Arnoldi Method

The Arnoldi method is a Krylov-based projection algorithm that computes an orthogonal basis of the Krylov subspace and at the same time computes a projected matrix. It is a generalization of the Lanczos method [62] and it was first introduced as a direct algorithm for reducing a general matrix into upper Hessenberg form. Later, it was discovered that this algorithm leads to a good iterative technique for approximating eigenvalues and corresponding eigenvectors of large sparse non symmetric matrices. It is really useful for cases when the matrix A is large but the matrix-vector products are relatively inexpensive to perform. Here, we begin with the basic definition of Arnoldi factorization and then describe a number of variations of it.

Definition 3.1 *An m-step Arnoldi factorization of $A \in \mathbb{C}^{n\times n}$ is defined as a relationship of the form*

$$AV_m = V_m H_m + f_m e_m^T,$$

where V_m is an $n \times m$ orthonormal matrix, H_m is a $m \times m$ upper Hessenberg matrix with non-negative sub diagonal elements, and $V_m^H f_m = 0$.

When the matrix A is Hermitian, this relationship is called an *m-step Lanczos factorization*, and the upper Hessenberg matrix H is actually real, symmetric, and tridiagonal.

Note that if (x, θ) is an eigenpair of H_m. Then, $x = V_m y$ satisfies the relationship

$$\begin{aligned} ||Ax - x\theta|| &= ||AV_m y - V_m y\theta|| \\ &= ||(AV_m - V_m H)y|| \\ &= ||f_k e_m^\top y|| \\ &= \beta|e_m^\top y|, \end{aligned}$$

where $\beta = ||f_m||$. The basic idea behind the Arnoldi factorization is to compute eigenpairs of the large matrix A from the eigenpairs of the small matrix H. Since we assume that $m << n$,

the eigenpairs of H can be computed by conventional means. The main purpose is to drive $|e_m^T y| \to 0$, so that the eigenpairs of H_m well approximates the eigenpairs of A. Here, the term $\beta|e_m^T y|$ is called the Ritz estimate, and describes the goodness of the eigenpair approximation. Of course, when $f_m = 0$ the equation will be simplified to

$$AV_m = V_m H_m$$

implying that V_m is an invariant subspace of A and the eigenpairs of H_m will be precisely m eigenpairs of A [54, 56, 62, 64]. The basic algorithm for Arnoldi factorization [54] is

Algorithm 1 m-step Arnoldi Factorization

Input:(V_m, H_m, f_m) such that $AV_m = V_m H_m + f_m e_m^T$

Output:$(V_{m+1}, H_{m+1}, f_{m+1})$ such that $AV_{m+1} = V_{m+1} H_{m+1} + f_{m+1} e_{m+1}^T$

1. $\beta_m = \|f_m\|; \qquad \upsilon \leftarrow f_m/\beta_m$;
2. $V_{m+1} \leftarrow (V_m, \upsilon); \qquad H_{m+1} \leftarrow \begin{pmatrix} H_m \\ \beta_m e_m^T \end{pmatrix}$;
3. $z \leftarrow A\upsilon_{m+1}$;
4. $h_{m+1} \leftarrow V_{m+1}^T z; \qquad f_{m+1} \leftarrow z - V_{m+1} h_{m+1}$;
5. $H_{m+1} \leftarrow (H_{m+1}, h_{m+1})$;
6. end;

In this algorithm a subindex is used in V and H to make their dimension explicit. At each iteration of the algorithm a new column of both H and V is computed and the explicit orthogonalization operations are carried out by means of Gram-Schmidt procedure [23].

It should be clear that Arnoldi factorization entirely depends on the choice of the starting vector. In fact, the factorization is uniquely determined by the choice of starting vector until a sub diagonal element of H_m is zero. At this point an invariant subspace has been computed and the factorization is continued with a new starting vector.

For good approximations, the starting vector used to begin the Arnoldi factorization should be rich in the direction of wanted eigenvectors and with very small components in the direction of other eigenvectors. However, in practice this is usually not possible and it likely requires many iterations. This causes serious problems because increasing number of iterations implies a growth in storage requirements and, more importantly, a growth of computational cost per

iteration. A good way of preventing this drawback is to restart the algorithm, that is, stop the algorithm after m iterations and rerun it with a new vector computed from the recently obtained spectral approximations. One possible approach is called explicit restarting. The idea of explicit restarting is to iteratively compute different m-step Arnoldi factorizations with successively better starting vectors. The starting vector for the next Arnoldi run is computed from the information available in the most recent factorization. The simplest way to select the new starting vector is to take the Ritz vector (or Schur vector) associated with dominant eigenvalue. This strategy is described in [23, 35].

There is another approach called implicit restarting. Implicit restarting combines the implicitly shifted QR mechanism with a m-step Arnoldi factorization. A m-step Arnoldi factorization is extended to a $(m + p)$-step Arnoldi factorization, which is then compacted again to an m-step one. The process of extending this new m-step factorization to a $(m + p)$-step factorization is iterated by applying shifts and condensing. The payoff this process is that each of these shifts results in the implicit application of a p th degree polynomial in A to the starting vector. The roots of this polynomial are the shifts that were applied to QR factorization. So, if we choose the shifts σ_i as the eigenvalues that are unwanted, the starting vector will be rich in the direction of wanted eigenvectors.

There are many strategies for selecting shifts σ_i. One of the useful strategies is the Exact Shift Strategy. This method takes the shifts as p eigenvalues of H_m which are furthest away from the wanted eigenvalues. However, there are alternative strategies such as Chebyshev polynomials [52] or Leja points [64].
Now, we are ready to present the full implicitly restarted Arnoldi method [54].

Algorithm 2 Implicitly Restarted Arnoldi Method Algorithm

Input: The matrix A, the number of eigenpairs to be computed m, the number of implicit shifts to apply to the factorization at each iteration p, a sort criterion which determines which are 'wanted' eigenvalues S, starting vector υ_0, tolerance τ

Output: $(x_1, \lambda_1), (x_2, \lambda_2), \ldots, (x_n, \lambda_n)$, approximations to the k wanted eigenvalues of A.

1. Using υ_0 as a starting vector, generate a m-step Arnodi factorization

$$AV = VH + fe_m^T$$

2. for $i = 1, 2, \ldots$ until $||Ax_i - \lambda_i x_i|| < \tau$ for all $i = 1, \ldots, m$

a.) Extend a m-step Arnoldi factorization to $(m + p)$ step Arnoldi factorization

$$AV = VH + fe_{m+p}^T$$

b.) Let $q = e_{m+p}$

c.) Sort the eigenvalues of H from best to worst according to the sort criterion S and take $\sigma_1, \ldots, \sigma_p$ to be the p worst eigenvalues.

d.) for $j = 1, 2, ..., p$

d1. Factor $H - \sigma_j I = QR$.

d2. $H \leftarrow Q^H HQ$.

d3. $V \leftarrow VQ$.

d4. $q \leftarrow q^H Q$.

e.) $f \leftarrow V(:, m + 1)\dot{H}(m + 1, m) + f\dot{q}(m)$.

f.) Take the first k columns on each side of he factorization to get $V = V(:, 1 : m)$, $H = H(1 : m, 1 : m)$.

g.) Take as eigenpair approximations (x_i, λ_i) the Ritz pairs of the problem.

3. end.

3.2.2 Krylov-Schur Method

To overcome the difficulties with implementing implicit restarting technique, the Krylov-Schur method was proposed by Stewart in 2001 [68].

The Krylov-Schur method is defined by generalizing the m-step Arnoldi factorization,

$$AV_m = V_m H_m + f_m e_m^T$$

computed by Algorithm 1, to a so-called Krylov decomposition of order m

$$AV_m = V_m B_m + \upsilon_{m+1} b_{m+1}^T, \tag{3.2}$$

where the matrix B_m is not restricted to be upper Hessenberg and b_{m+1} in an arbitrary vector. Assume that all the υ_i vectors are mutually orthonormal. Then premultiplying Equation (3.2) by V_m^T shows that B_m is the Rayleigh quotient $V_m^T A V_m$. So, the Rayleigh-Ritz procedure is still valid.

The last equation can be written in the form

$$AV_m = \begin{bmatrix} V_m & \upsilon_{m+1} \end{bmatrix} \begin{bmatrix} B_m \\ b_{m+1}^T \end{bmatrix}. \tag{3.3}$$

A special case of the above relation is called the Krylov-Schur decomposition, in which matrix B_m is in real Schur form, that is, quasi-triangular form displaying eigenvalues are in the 1×1 or 2×2 diagonal blocks.

It can easily be seen that Arnoldi decomposition is a special case of the Krylov decomposition. It is shown in [68] that any Krylov decomposition is equivalent to an Arnoldi decomposition, that is, both have the same Ritz approximations. Also, a Krylov decomposition can be transformed into an equivalent Krylov-Schur decomposition by means of orthogonal similarity transformations. The main idea of the Krylov-Schur method is to expand iteratively with the Arnoldi process and contract a Krylov-Schur decomposition. A systematic description of the method is given in the following algorithm that can be found in [36].

Algorithm 3 Krylov-Schur Method

Input: Matrix A, starting vector x_1, and number of steps m;

Output: $m \leq p$ Ritz pairs

1. Build an initial Krylov decomposition of order m
2. Apply orthogonal transformations to get a Krylov-Schur decomposition
3. Reorder the diagonal blocks of the Krylov-Schur decomposition
4. Truncate to a Krylov-Schur decomposition of order p
5. Extend to a Krylov decomposition of order m
6. If not satisfied, go to step 2

Step 1 can be accomplished with an Arnoldi decomposition. At step 2, to get a Krylov-Schur decomposition it is necessary to apply the QR algorithm in order to compute an orthogonal matrix Q_1 such that $T_m = Q_1^T B_m Q_1$ which has real Schur form, and then

$$AV_m Q_1 = \begin{bmatrix} V_m Q_1 & \upsilon_{m+1} \end{bmatrix} \begin{bmatrix} T_m \\ b_{m+1}^T Q_1 \end{bmatrix}. \tag{3.4}$$

At this point, from the diagonal blocks of T_m the Ritz values are already available. These Ritz values are divided in two subsets. The first one contains $p < m$ "wanted" Ritz values and the other one contains $m - p$ "unwanted" Ritz values. By step 3 the algorithm moves the subset of wanted Ritz values to the leading principal submatrix of T_m. This can also be accomplished by means of an orthogonal transformation Q_2, resulting in the following reordered Krylov-Schur decomposition:

$$A\tilde{V_m} = \begin{bmatrix} \tilde{V_m} & \upsilon_{m+1} \end{bmatrix} \begin{bmatrix} T_w & * \\ 0 & T_u \\ b_w^T & * \end{bmatrix}, \tag{3.5}$$

where $\tilde{V_m} = V_m Q_1 Q_2$, $\lambda(T_w) = \Omega_w$. Here, Ω_w shows the set of wanted Ritz values, $\lambda(T_u) = \Omega_u$ where Ω_u is a set of unwanted Ritz values, and b_w^T is the length-p leading sub vector of $b_{m+1}^T Q_1 Q_2$. The truncation in step 4 of the algorithm is achieved simply by writing

$$A\tilde{V}_p = \begin{bmatrix} \tilde{V}_p & \tilde{\upsilon}_{p+1} \end{bmatrix} \begin{bmatrix} T_w \\ b_w^T \end{bmatrix}, \tag{3.6}$$

where $\tilde{V}_p$ is equal to the first p columns of $\tilde{V}_m$, but $\tilde{\upsilon}_{p+1} = \upsilon_{m+1}$. Finally, step 5 is accomplished by means of a variation of Algorithm 1 in which the vectors are computed starting from υ_{p+2} but vectors υ_1 to υ_{p+1} are also taken into account in the orthogonalization step.

If the total number of works are compared in implicitly restarted Arnoldi method (IRA) and Krylov-Shur method, IRA is superior to Krylov-Schur in marginal operation cost [68]. Moreover, implicitly restarted Arnoldi method with exact shifts and Krylov-Schur method with the same shifts have the same effect which was proved in [68]. But each have theirs pros and cons. Different polynomials can be used instead of shifts in IRA but this is not possible in Krylov-Schur. On the other hand when exact shifts are used Krylov-Schur is more preferable due to the reliable process for exchanging the eigenvalues.

In the case of that the matrix A is symmetric, the Krylov-Schur method is equivalent to another method called the thick-restart Lanczos method [72]. The most important difference between the symmetric and non symmetric cases of the Krylov-Schur method is the shape of matrix B_m. The structure of this matrix B_m has more difficulties than in the case of a simple explicitly restarted variant of Lanczos, where the projected matrix consisted of a diagonal part and a tridiagonal part. In the case of thick-restart Lanczos, obtaining the Krylov-Schur form of B_m is equivalent to diagonalizing it. So, taking into account its special structure may yield a reduction in the operation count for standard solvers. Therefor, using a general dense symmetric method is more appropriate. An alternative may be to write a specific algorithm for this particular structure, although it is not obvious how to do this.

The other thing to be considered in the symmetric case is whether orthogonalization of the Lanczos vectors should be carried out explicitly against all previous vectors or using a technique for the cure of loss of orthogonality. In the context of thick-restart Lanczos, full orthogonalization is the best option since it provides the maximum robustness.

3.2.3 Augmented Block Householder Arnoldi Method

The implicitly restarted Arnoldi (IRA) method proposed by Sorensen [62, 64] modifies the starting vector at each iteration via shifted *QR* algorithm as we presented previously. This method is based on a very popular software package ARPACK [44]. A comparison of softwares provided in [42] concluded that ARPACK was generally the fastest and most dependable. However, many numerical examples have shown that there may be propagated round-off errors which can delay or prevent convergence of desired eigenvalues and eigenvectors [42]. The reason for this numerical instability is the underlying *QR*-algorithm. It is shown in [47] that the implicitly restarted Arnoldi (IRA)method of Sorensen [64] can be implemented by augmenting the sequence of Krylov subspace basis by certain Ritz vectors. This implementation is mathematically equivalent to the implementation in [64]. It can be less sensitive to propagated round-off errors than [64]. This relationship has been exploited in [72] for symmetric eigenvalue problems, in [46] for linear systems and non symmetric eigenvalue problems and in [4] for singular value problems by Baglama and Reichel. The extension of this idea to block Krylov subspaces (3.8) has been implemented by Möller for non symmetric eigenvalue problems. The extensions to block methods have many favorable attributes even though they are not mathematically equivalent to the block form of the IRA method.

The main purpose of augmented block Householder Arnoldi method is creating an augmented block Krylov subspace method for the eigenvalue problem

$$Ax = \lambda x. \tag{3.7}$$

The block Arnoldi method only differs from the Arnoldi method in that it uses a set of starting vectors $X = [x_1, x_2, x_3, \ldots, x_n]$ and builds an orthonormal basis for the block Krylov subspace

$$K_{mr}(A, X) = \text{span}\left\{X, AX, A^2X, \ldots, A^{m-1}X\right\}. \tag{3.8}$$

A block routine generally requires more computational effort and larger subspaces for good approximations. However, a block routine is more efficient to compute multiple or clustered eigenvalues than an unblock routine [1, 2, 43]. This advantage of block routines has resulted in a considerable number of softwares/algorithms. The augmented block Householder Arnoldi

(ABHA) method developed by Baglama [3] combines the advantages of a block routine and an augmented routine. The development of an augmented block Arnoldi method is not new. The implementation of it for solving linear systems of equations is presented in [48] and for solving non symmetric eigenvalue problems in [49]. However, here we will present a new implementation of augmented block Householder Arnoldi method with MATLAB code *ahbeigs* which is introduced in Chapter 4.

The foundation of ABHA method is the use of Householder process to create an orthonormal basis for the block Krylov subspace (3.8). Algorithm 4 [3] extends the Householder Arnoldi method in [69] to block form. It uses the Householder process for creating an orthonormal basis for the Krylov subspace. This method uses the compact WY representation of the Householder product. In this representation the Q matrix in the Householder QR-decomposition is formed from the product with the form $I + YTY^T$, where Y is a lower trapezoidal matrix and T is a square upper triangular matrix.

After applying these transformations to the subspace $K_{mr}(A, X)$, the following block Arnoldi decomposition is obtained:

$$AV_{mr} = V_{mr}H_{mr} + V_{(m+1)}H_{(m+1,m)}E_r^T, \tag{3.9}$$

where V_{mr} is an $n \times mr$ orthogonal matrix and H_{mr} is $mr \times mr$ upper block Hessenberg matrix. During the computation of the block Arnoldi method $H_{j+1,j}$ which is a sub-diagonal block of the blocked Hessenberg matrix may become singular. This means that a set of vectors in (3.8) have become linearly dependent on previously computed vectors. Different from the single vector Arnoldi method, this linear dependency of vectors may not imply an invariant subspace has been computed unless $H_{j+1,j} \equiv 0$. This breakdown rarely occurs, and the Householder block Arnoldi method handles this in step 3 of the Algorithm 4 by adding a random vector at step 3 so that a valid QR factorization is computed. Now, we are ready to outline our block restarted method.

After an m-step of Arnoldi method the real Schur decomposition of H_{mr} is computed such that

Algorithm 4 [3] Block Arnoldi Householder Algorithm

Input: $A \in \mathbb{R}.^{n \times n}$

Output: $Y \in \mathbb{R}^{n \times mr+r}$, $T \in \mathbb{R}^{mr+r \times mr+r}$, and $H_{i,j} \in \mathbb{R}^{r \times r}$, where $j = 0, \ldots, m$, $i = 1, \ldots, j+1$

1. Choose r random vectors x_i and set $X = [x_1, \ldots, x_r] \in \mathbb{R}^{n \times r}$;

2. For $j = 1, 2, \ldots, m$

3. Compute the Householder QR-decomposition where

$X(jr+1 : n, 1 : R) = QR$ and $Q = (I + WSW^T)\begin{bmatrix} I \\ 0 \end{bmatrix}$, where $I \in \mathbb{R}^{r \times r}$ and $0 \in \mathbb{R}^{(n-r) \times r}$

4. if $j = 0$

a.) Set $\begin{Bmatrix} Y & := & W \\ T & := & S \\ H_{(1,0)} & := & R \end{Bmatrix}$

else

b.)Set $\left\{ \begin{bmatrix} H_{(1,j)} \\ \vdots \\ H_{(j,j)} \end{bmatrix} := \begin{bmatrix} X(1 : jr, 1 : r) \end{bmatrix}, \begin{bmatrix} H_{(j+1,j)} \end{bmatrix} \quad \begin{bmatrix} R \end{bmatrix} \right\}$, where $X(1 : jr, 1 : r) \in \mathbb{R}^{jr \times r}$ and $R \in \mathbb{R}^{r \times r}$

c.) Set $\begin{Bmatrix} W := \begin{bmatrix} 0 \\ W \end{bmatrix} \text{where} 0 \in \mathbb{R}^{jr \times r} \text{and} W \in \mathbb{R}^{n-jr \times r} \\ T := \begin{bmatrix} T & TY^TWS \\ 0 & S \end{bmatrix} \in \mathbb{R}^{(j+1)r \times (j+1)r} \\ Y := \begin{bmatrix} Y & W \end{bmatrix} \in \mathbb{R}^{n \times (j+1)r} \end{Bmatrix}$

end

5. If $j < m$

6. Compute $X = (I + YT^TY^T)A(I + YTY^T)\begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix}$ where $0 \in \mathbb{R}^{jr \times r}$, $I \in \mathbb{R}^{r \times r}$ and

$0 \in \mathbb{R}^{(n-jr-r) \times r}$, respectively.

7. end

8. end

$$H_{mr} Q_{mr}^{(H_{mr})} = Q_{mr}^{(H_{mr})} U_{mr}^{(H_{mr})}, \tag{3.10}$$

where $U_{mr}^{(H_{mr})}$ is a quasi-triangular matrix where the eigenvalues of H_{mr} are either a real 1×1 matrix or a real 2×2 matrix. The latter case constitutes complex conjugate pairs and $Q_{mr}^{(H_{mr})} = [q_1^{(H_{mr})} \dots q_{mr}^{(H_{mr})}]$ is an orthogonal matrix. The real Schur decomposition can be reordered so that the desired eigenvalues occur in the upper left part of the matrix $U_{mr}^{(H_{mr})}$.

Assume that k be the number of desired eigenvalues and for ease of the presentation, k does not split a conjugate pair. After reordering the real Schur decomposition of H_{mr} and truncating the last $(mr - k)$ columns we have

$$H_{mr} Q_k^{(H_{mr})} = Q_k^{(mr)} U_k^{(mr)}, \tag{3.11}$$

For a given matrix A we can find the approximate real partial Schur decomposition $AQ_k^{(A)} = Q_k^{(A)} U_k^{(A)}$ from the equation (3.10), where

$$Q_k^{(A)} = [q_1^{(A)}, q_2^{(A)}, \dots, q_k^{(A)}] = V_{mr} Q_k^{(H_{mr})} \quad and \quad U_k^{(A)} = U_k^{(H_{mr})}. \tag{3.12}$$

We can easily obtain the partial eigenvalue decomposition of A by computing the eigenvalue decomposition $U_k^{(H_{mr})} S_k = S_k D_k^{(H_{mr})}$ and setting

$$V_k^{(A)} = V_{mr} Q_k^{(H_{mr})} S_k, D_k^{(A)} = D_k^{(H_{mr})} \quad to \quad get \quad AV_k^{(A)} = V_k^{(A)} D_k^{(A)}. \tag{3.13}$$

Using the (3.9), (3.10) and (3.13) we have the following

$$AV_k^{(A)} - V_k^{(A)} D_k^{(A)} = V^{(m+1)} H_{(m+1,m)} E_r^T Q_k^{(H_{mr})} S_k. \tag{3.14}$$

It can be easily seen from (3.14) that we have an acceptable approximate partial eigenvalue decomposition of A when

$$\|H_{(m+1,m)} E_{r^T} Q_k^{(H_{mr})} S_k\| \leq \alpha tol, \tag{3.15}$$

where tol is a user input tolerance value and α is chosen to assure a small backward error, see [3], [44]. Then, rearranging (3.14) gives the following relationship

$$AQ_k^A = \begin{bmatrix} Q_k^A & V_{m+1} \end{bmatrix} := \begin{bmatrix} U_k^A \\ H_{(m+1,m)} E_r^T Q_k^{H_{mr}} \end{bmatrix}. \tag{3.16}$$

By means of orthogonal transformations (3.16) can be transformed into a k block Arnoldi decomposition. Therefore, the block Arnoldi algorithm can be restarted or continued with the matrix $V_{(m+1)}$. In order to continue block Arnoldi Householder Algorithm 3, the orthogonal matrix $\begin{bmatrix} Q_k^{(A)} & V^{(M+1)} \end{bmatrix}$ is replaced into the compact WY representation of the Householder product. The reference [3] can be seen more details. After this orthogonalization process we have

$$(I + \bar{Y}\bar{T}\bar{Y}^T) I_{k+r} = \begin{bmatrix} Q_k^A & V_{m+1} \end{bmatrix} R, \tag{3.17}$$

where R is a diagonal matrix of ± 1. Then multiplying (3.16) by $\hat{R} = R_{(k,k)} \in \mathbb{R}^{k \times k}$ gives

$$AQ_k^{(A)} \hat{R} = (I + \bar{Y}\bar{T}\bar{Y}^T) I_{k+r} \bar{H}_{k+r}. \tag{3.18}$$

If we choose $Q_k^{(A)} \hat{R} = [q_1, q_2, ..., q_k]$, the block Householder Algorithm can be continued with the next set of r vectors. Then, matrices $\bar{Y}$ and $\bar{T}$ are updated to get the next set of vectors and the restarted method is continued.

The following algorithm combines the results and restarted method to obtain augmented block Householder Arnoldi method [3]:

3.3 Jacobi-Davidson Type Methods

This section is based on books [23, 63, 67] and the articles [60, 61].

Jacobi-Davidson methods [60, 61] have been introduced as a powerful technique for solving a variety of eigen problems. The basic idea in these methods is projecting the matrix onto a

Algorithm 5 Augmented Block Arnoldi Householder Algorithm

Input: $A \in \mathbb{R}^{n\times n}$, k, m, r, tol such that $(m-1)r \geq k$

Output:eigenvalues $\lambda_{j}{}_{j=1}^{k}$ and eigenvectors $x_{j}{}_{j=1}^{k}$ of A

1. Perform m steps of the block Arnoldi Householder algorithm 3 to get the block Arnoldi decomposition (3. 3)

2. Compute and start the Real Schur decomposition of the matrix H_{mr} and $\overline{H}_{k+r+m_1 r+r}$

3. Check the convergence of the k desired eigenvalues using

$$||H_{(m+1,m)}E_{r^T}Q_k^{(H_{mr})}S_k|| \leq \alpha \dot{tol}$$

a.)if all k values converge then compute $\lambda_{j}{}_{j=1}^{k}$ and $x_{j}{}_{j=1}^{k}$ using (3. 7)

4. Compute restarting vectors $(I + \overline{YTY}^T)I_{k+r} = [Q_k^{(A)}V_{(m+1)}]R$

5. Compute the matrix $\overline{H}_{k+r,k}$

6. Perform m_1 steps of the block Arnoldi Householder Algorithm to get the block Arnoldi decomposition

7. Go to step 2.

subspace like the Krylov subspace algorithms. Different from the methods like Arnoldi and Krylov-Schur depending on preserving Krylov subspace, the Jacobi-Davidson method does not insist on keeping a Krylov structure in the projected subspace.

Jacobi-Davidson methods are based on the approach by Jacobi and Davidson. In the Davidson method [60], the subspace is expanded by orthogonalizing the correction against the residual. This method works very well with the matrices which are diagonally dominant. Jacobi method for eigenvalue approximation is a combination of Jacobi rotations, Gauss-Jacobi iterations and an almost forgotten method called Jacobi's orthogonal component correction (JOCC) [39]. Combining the ideas in the Jacobi and Davidson methods, a new approach called Jacobi-Davidson method is introduced in [60] for symmetric matrices.

Here, we introduce the Jacobi-Davidson type algorithms for non symmetric eigenvalue problems [61]. These algorithms are based on the Jacobi-Davidson method described in [60] and adapted for generalized and standard eigenvalue problems. In these algorithms the Jacobi-Davidson approach is modified such that partial (generalized) Schur forms are computed.Since they involve orthogonal bases, the partial Schur forms have been chosen for numerical stability.

In Jacobi-Davidson methods, the small projected problem is reduced to Schur form by the *QZ* method [45] for generalized eigen problems or by *QR* method for the standard eigen problems. The construction of the subspace and the projected system is viewed as iterative inexact forms of *QZ* and *QR* methods. For this reason the methods introduced here have been named JDQZ and JDQR, respectively. The JDQZ method produces a partial generalized Schur form for the generalized eigenvalue problem and JDQR generates a partial Schur form for the standard eigenvalue problem. Here we focus on JDQR method, however, a brief introduction for JDQZ method is presented as JDQR method is a simplification of the JDQZ method.

Similar to subspace approaches for standard eigenvalue problems, the approximate eigenvector $\tilde{q}$ is selected from a search subspace span$\{V\}$ in each step of the method. The Galerkin condition, with associated approximate generalized eigenvalue $\left\langle \tilde{\alpha}, \tilde{\beta} \right\rangle$, involves orthogonality with respect to some test subspace span$\{W\}$

$$\tilde{\beta}A\tilde{q} - \tilde{\alpha}B\tilde{q} \perp \text{span}W \tag{3.19}$$

For generalized eigenvalue problems, it is natural to take the test subspace span$\{W\}$ different from the search subspace called the Petrov-Galerkin approach. Assume that search subspace and test subspace are of the same dimension, say j. Equation (3.19) leads to the projected eigen problem

$$(\tilde{\beta}W^T AV - \tilde{\alpha}W^T BV)u = 0, \tag{3.20}$$

such that it can be easily solved by standard techniques, and a solution $(u, \left\langle \tilde{\alpha}, \tilde{\beta} \right\rangle)$ is selected. Then the Petrov vector $\tilde{q} = Vu$ and the residual $r \equiv \tilde{\beta}A\tilde{q} - \tilde{\alpha}B\tilde{q}$ associated with the Petrov value $\left\langle \tilde{\alpha}, \tilde{\beta} \right\rangle$ are computed. In each step of the iterative process, the subspaces span$\{V\}$ and span$\{W\}$ are expanded. In the Jacobi-Davidson method introduced here, the search subspace is expanded by a vector v such that it is orthogonal to $\tilde{q}$ and it solves approximately the Jacobi correction equation

$$(I - \frac{\tilde{z}\tilde{z}^T}{\tilde{z}^T\tilde{z}}) - (\tilde{\beta}A - \tilde{\alpha}B)(I - \frac{\tilde{q}\tilde{q}^T}{\tilde{q}^T\tilde{q}})v = -r. \tag{3.21}$$

In the next step of the iteration process of the algorithm span$\{V, v\}$ defines the new search

subspace. As we choose orthogonal matrices V and W, the columns of V and W are orthonormalized by modified Gram-Schmidt method. Then the QZ method is used to reduce Equation (3.20) to a partial generalized Schur form. After convergence, the partial Schur form is expanded with the converged Schur vector, and the algorithm is repeated with a deflated pencil for other eigenpairs. More details about $JDQZ$ method can be found in [61].

Now, we are ready to introduce a simple case of this method called Jacobi-Davidson type QR (JDQR) method.

3.3.1 Jacobi-Davidson QR Method

Jacobi-Davidson type QR method is a simplification of Jacobi-Davidson type QZ method [61] to standard eigenvalue problem. In Jacobi-Davidson method for standard eigenvalue problems, the projected eigenproblem reduce to

$$(V^*AV - \tilde{\lambda}V^*V)u = 0. \tag{3.22}$$

For this low-dimensional problem a solution, say $(u, \tilde{\lambda})$, is selected by standard computational techniques. The Ritz value λ and the Ritz vector $\tilde{q} \equiv Vu$ form an approximate eigenvalue and eigenvector with residual $r \equiv (A - \tilde{\lambda}I)\tilde{q}$.

For the expansion of V, a vector v that is orthogonal to $\tilde{q}$ is taken and it solves the Jacobi correction equation

$$\tilde{q}^*v = 0 and (I - \tilde{q}\tilde{q}^*)(A - \tilde{\lambda}I)(I - \tilde{q}\tilde{q}^*)v = -r. \tag{3.23}$$

The expanded search subspace is span $\{V, v\}$ where V is a orthonormal matrix, i.e., $V^*V = I$. In this method for the construction of an orthonormal basis of the search subspace modified Gram-Schmidt is used.

If $\tilde{\lambda}$ is replaced in the correction equation (Equation (3.23)) by an eigenvalue λ, then the space spanned by V and the exact solution of the Jacobi correction equation contains the associated eigenvector.

The projected eigenproblem (3.22) is reduced to Schur form by the QR algorithm and then the Schur form is exploited for the selection of a Ritz pair $(\tilde{q}, \tilde{\lambda})$ and for restriction of the dimension of the subspace span $\{V\}$.

Now, we will give the main expressions for Jacobi-Davidson type QR algorithm based on the article [61].

Assume that we have detected the $(k-1)$ Schur pairs, that is, we already know the partial Schur

$$AQ_{k-1} = Q_{k-1}R_{k-1}.$$

Then the new Schur pair (q, λ) is an eigenpair of the deflated matrix

$$(I - Q_{k-1}Q_{k-1}^T)A(I - Q_{k-1}Q_{k-1^T}). \tag{3.24}$$

The eigenvalue problem for the deflated matrix (3.20) can be solved easily. For the deflated matrix (3.20) Jacobi-Davidson method constructs a subspace span $\{V\}$ for finding the approximate eigenpairs, where V is an orthonormal matrix such that $V^T Q_{k-1} = 0$. Then, for the deflated interaction matrix M we obtain

$$M \equiv V^T(I - Q_{k-1}Q_{k-1}^T)A(I - Q_{k-1}Q_{k-1^T})V = V^T AV. \tag{3.25}$$

For a wanted eigenpair of the deflated matrix (3.20), the ordered Schur form

$$MU = US$$

gives an approximated eigenpair $(\tilde{q}, \tilde{\lambda}) \equiv (VU(:,1), S(1,1))$. Then, according to the Jacobi-Davidson approach, the search subspace span $\{V\}$ is expanded by the orthonormal complement of v to V, where v is the approximate solution of the deflated Jacobi correction equation

$$Q_{k-1}^T v = 0, \tilde{q}^T v = 0$$

and

$$(I - \tilde{q}\tilde{q}^T)(I - Q_{k-1}Q_{k-1}^T)(A - \tilde{\lambda}I) \times (I - Q_{k-1}Q_{k-1}^T)(I - \tilde{q}\tilde{q}^T)v = -r, \tag{3.26}$$

where $r \equiv (I - Q_{k-1}Q_{k-1}^T)(A - \tilde{\lambda}I)(I - Q_{k-1}Q_{k-1}^T)\tilde{q}$.

Note that the projections in Equation (3.26) can be subdivided into two parts.The first part is $(I - \tilde{q}\tilde{q}^T)$ associated with Jacobi-Davidson and the second one is the deflation part $(I - Q_{k-1}Q_{k-1}^T)$.

Similar to Arnoldi's method, for subspace iterations there is two deflation techniques in the literature. They are called as explicit and implicit deflation techniques. In explicit deflation, after detection of Schur vector the computation is continued with a deflated matrix. In implicit deflation, each new vector is generated with A itself for the search subspace. Then it is made orthogonal to the detected Schur vectors before adding it to the search subspace. This method uses a mixture of both techniques. In Jacobi correction equation

$$\tilde{q}^T v = 0 \quad \textit{and} \quad (I - \tilde{q}\tilde{q}^T)(A - \tilde{\lambda}I)(I - \tilde{q}\tilde{q}^T)v = -r$$

with *residual* $r \equiv (A - \tilde{\lambda}I)\tilde{q}$, the explicitly deflated matrix can be used. The solutions of the deflated correction equations are orthogonal to the detected Schur vectors. So, there is no need to use deflated matrix for computing the deflated interaction matrix M. It is also possible to use implicit deflation in the following way: the correction equation with the non deflated A is solved approximately and the resulting solution is made orthogonal to the detected Schur vectors. In this approach the expensive matrix-vector multiplications are avoided but, explicit deflation seems to improve the condition numbers the linear system and this leads to a faster convergence process for the Jacobi correction equation. When compared with implicit deflation, it seems that the explicitly deflated correction Equation (3.26) leads to more stable results. This can be explained as follows. The resulting solution of the correction equation without deflation may have an important component in the space spanned by the detected Schur vectors. Subtracting this component as an implicit deflation may result in cancellation. So, working with an explicitly deflated matrix prevents this cancellation.

Now, we briefly discuss preconditioning for the Jacobi correction equation. We need to solve a deflated Jacobi correction equation (3.26) for a given $\tilde{q}$ and $\tilde{\lambda}$ in each iteration step. For

the approximate solution of this equation a Krylov subspace method may be used. The rate of convergence and the efficiency of Krylov subspace methods can be improved by preconditioning. However, there may be a problem during the identification of an effective preconditioner. For instance, for interior eigenvalues the construction of an effective incomplete LU-factorization for $\left(A - \tilde{\lambda}I\right)$ may require much fill in, which makes the construction expensive. So, it may be a good strategy to compute a good (and possibly expensive) preconditioner K for $(A - \tau I)$ where τ is an user specified fixed target value and to use

$$\tilde{K} \equiv (I - \tilde{q}\tilde{q}^T)(I - Q_{k-1}Q_{k-1}^T)K(I - Q_{k-1}Q_{k-1}^T)(I - \tilde{q}\tilde{q}^T) \tag{3.27}$$

as the preconditioner for various $\tilde{q}$ and $\tilde{\lambda}$. For more details about preconditioning in JDQR method see [61].

JDQR algorithm has some nice properties. While the process converges to a Schur pair, the search subspace V will provide good initial approximations for the nearby Schur pairs. Moreover, slow convergence during one stage may be compensated for by faster convergence in the next stage, because the subspace span$\{V\}$ will be enriched with more components of other Schur pairs due to the repeated amplifications.

CHAPTER 4

EIGENVALUE SOLVERS

In the chapter, we present description of programs and software packages described in Chapter 3. Since eigenvalue problems have a wide range of application areas in science and engineering, there exists many efficient eigensolvers. Most of the methods for sparse non symmetric eigenvalue problems were developed with in the last fifteen years. Parallel too the development of new methods, several eigenvalue solvers were developed such that most of them can be used in applications. For a detailed survey of current solvers you can see [34].

The programs used in this study are written in MATLAB (speig, ahbeigs, jdqr) and C++ (SLEPc). MATLAB is a numerical computing environment which has good visualization skills. Today, it is a popular language among numerical mathematicians. On the other hand, C++ is a general purpose programming language. It is used in both industrial and academic environment. All of the eigenvalue solvers investigated in this study can be found on the Internet freely. They can only be applied to sparse matrices. The solvers implemented in MATLAB are designed for serial computation, however, the SLEPc is also designed for parallel computation. In this study, we use both programming languages in serial computation.

4.1 Descpription of speig

speig is a MATLAB implementation of implicitly restarted Arnoldi method (IRAM) described in Chapter 2. Its name is built by the combination of the first letters of words sparse and eigenvalue. It is developed as an alternative to the *eig* command of MATLAB which is for dense matrices. The *speig* solver is available with its own package at ftp.task.gda.pl/pub/software/matlab/toolbox/matlab/sparfun/speig/. A detailed description and implementation of the solver

$'LM'$	Largest Magnitude
$'SM'$	Smallest Magnitude
$'LR'$	Largest Real Part
$'SR'$	Smallest Real Part
$'BE'$	Both ends

Table 4.1: Character signs for σ

can be found at [54].

The basic syntax of the solver *speig* is

$$>> d = speig(A),$$

such that it provides one output argument the vector d which is the eigenvalues of A. If the eigenvectors are also needed the syntax will be

$$>> [V, D] = speig(A)$$

such that it provides two output arguments, the matrix V consisting of eigenvectors and the diagonal matrix D with eigenvalues on its diagonal.

The general syntax of the eigenvalue solver *speig* is

$$>> d = speig(A, k, \sigma)$$

where A is the matrix whose eigenvalues are desired to compute, k is the desired number of eigenvalues and σ is the around which the user wants to compute the desired eigenvalues. Here, σ is numeric or a two letter string described in Table 4.1.

There are also many input parameters for the solver *speig* other than these ones. The default values of parameters in the solver *speig* can be changed with the help of two functions: *speigset* and *speigget*.

The function *speigset* changes and saves the parameter values by the syntax

Parameter Name	Description	Default value
n	Dimension of the problem	none
p	Dimension of Arnoldi basis	2k
tol	Tolerance for convergence of $\frac{\|AV-VD\|}{\|A\|}$	10^{-10} (symmetric A, B) 10^{-6} (non symmetric A, B)
maxit	Maximum number of Arnoldi iterations	300
issym	Positive if A is symmetric, 0 otherwise	0
dopoly	Positive if it specifies a matrix vector product, σ is LR, SR or numeric and polynomial interpolation is to be used to accelerate convergence	0
gui		0

Table 4.2: Parameter list of *speig*

$$>> opts = speigset(`name1', `value1', `name2', `value2', \ldots),$$

where *name1* represents the name of the parameter and *value1* represents the specified value of the parameter.

speigget is designed to extract the parameter values that are created by *speigset*. Its syntax is

$$>> v = speigget(opts,' name1').$$

The parameters and their default values for the solver *speig* are given in Table 4.2.

The parameter *dopoly* shows the activation or deactivation of polynomial acceleration. The *qui* parameter is related to opening a convergence history window of the solver. It shows the residual norm at each step, current iteration number, elapsed time and convergence report. The general properties of the problems like matrix size, tolerance and maximum number of iterations are also reported.

4.2 Descpription of ahbeigs

ahbeigs is a MATLAB implementation of the augmented block Householder Arnoldi method (AHBEIGS). Its name is built by the combination of the first letters of the algorithm augmented block Householder Arnoldi and eigenvalues. It is developed by J. Baglama to take

the advantage of block Krylov subspaces. It is used for both general and standard eigenvalue problems. The solver *ahbiegs* is available at www.math.uri.edu/~jbaglama/software. A detailed description can be found in [3].

The basic syntax of *ahbeigs* is

$$>> d = ahbeigs(A),$$

where d is a vector with entries consisting of desired eigenvalues of the matrix A which can be represented as a numeric or as a M-file ('Afunc'). If A is passed as an M-file, then the size of the matrix A must also be introduced. If the corresponding eigenvectors are needed, the syntax becomes

$$>> [X, D] = ahbeigs(A),$$

where D is a diagonal matrix that contains the desired eigenvalues along the diagonal and the matrix X contains the corresponding eigenvectors, such that $AX = XD$ or $AX = BXD$ for generalized eigenvalue problem.

It is also possible to see the convergence history of the algorithm by the following syntax:

$$>> [X, D, FLAG] = ahbeigs(A),$$

which returns the same as the above option plus a two dimensional array FLAG that reports if the algorithm converges and the number of matrix vector products.

Similar to other algorithms, *ahbiegs* is also contains many parameters and it is possible to change these parameter values by using 'OPTS' command of *ahbeigs*. The structure to change the default values of parameters is

>> OPTS.parameter name = desired value of the parameter

Then by adding the 'OPTS' command as an input argument to one of the syntaxes of *ahbeigs*,

Parameter	Description	Default value
k	Number of desired eigenvalues	6
adjust	Initial number of vectors added to the k restart vectors to speed up the convergence	3
tol	Tolerance used for convergence	10^{-6}
maxit	Maximum number of iterations	100
blsz	Block-size of block Arnoldi Hessenberg matrix, H_{mr}	3
nbls	Number of blocks in the block Arnoldi Hessenberg matrix, H_{mr}. If value of *nbls* is not sufficiently large enough then *ahbeigs* will not converge or miss some desired eigenvalues	10
dspr	Sets history	0
V_0	starting vector	$V_0 = randn$
υ_0	Initial matrix of r columns for the block Arnoldi method	rand

Table 4.3: Parameter list of *ahbeigs*

the user can achieve to set up new parameter values. Table 4.3 shows the parameters and default values used in the eigenvalue solver *ahbeigs*.

The parameter *blsz* shows the number of columns of starting matrix which is the r value in the formula. *nbls* parameter is the m value in the formula and if required it is increased automatically by the solver. The input parameter *sigma* shows the portion of the spectrum. It can be a numeric or a letter string defined in Table 4.1. The convergence of the method is determined by the parameter *tol* described in Chapter 3.

4.3 Description of jdqr

jdqr is a MATLAB implementation of Jacobi-Davidson QR method. The name JDQR comes from the first letters of the Jacobi-Davidson QR method. It is developed by Gerard Sleijpen and Han Der Vorst. A detailed description of the method can be found in [60, 61]. The MATLAB code of the solver is available at www.staff.science.uu.nl/~sleij101/JD_software.

The basic syntax of *jdqr* is given as

$$>> d = JDQR(A),$$

where d is a vector whose entries are eigenvalues of the matrix A. Unless it is changed, vector d includes five largest eigenvalues of the matrix A. When the corresponding eigenvectors are

needed, the syntax becomes

$$>> [X, Lambda] = JDQR(A).$$

The way of calculating other parts of the spectrum can be achieved by the command

$$>> d = JDQR(A, k, sigma)$$

where the first input argument is either a square matrix that can be full or sparse, symmetric or non symmetric, real or complex. k is the number of desired eigenvalues and σ is a numeric or a two letters string that determines the portion of the spectrum. The two letters string for *jdqr* is given in Table 4.1.

Other parameters in the solver can also be changed in a practical way by the following options structure:

$$>> \text{OPTS.parameter name} = \text{parameter value}.$$

Then assigning these changes to the code by

$$>> d = JDQR(A,OPTS)$$

ends up with changes of the parameter.

The parameters used in *jdqr* are given in Table 4.4.

The type of the Ritz values used during calculation is determined by parameter *test space*. If it is assigned as 'STANDARD', standard Ritz values are used. If it is assigned as 'HARMONIC', the harmonic Ritz values are selected. For computing interior eigenvalues, choosing the type of the Ritz values as 'HARMONIC' gives better results.

The parameter *LSolver* determines the opportunity to select the linear solver used for correction equation. Linear solver can be chosen as GMRS, CG, MINRES or many others which requires a positive preconditioner to work.

Parameter	Description	Default value
Tol	Convergence tolerance	1e-8
jmin	Minimum dimension search subspace	k+5
jmax	Maximum dimension search subspace	jmin+5
maxit	Maximum number of Arnoldi iterations	100
v0	Starting space	ones+1*rand
testspace	For using harmonic Ritz values. If $'testspace' = 'harmonic'$, then $SIGMA = 0$ is the default value for $SIGMA$	'Standard'
Lsolver	Linear solver	'GMRES'
LS_Tol	Residual reduction linear solver	1, 0.7, 0.7^2, ...
LS_Maxit	Maximum number iterations of linear solver	5
Precond	Preconditioner	M=[].

Table 4.4: Parameter list of *jdqr*

By assigning the parameter value of *disp* to 1, the user will have a chance to see convergence history and residual at each step. If it is not changed, there is no chance to see the history.

4.4 Descpription of SLEPc

SLEPc, the Scalable Library for Eigenvalue Problem Computations, is a software library for the solution of large, sparse eigenvalue problems. It is built on top of PETSc (Portable, Extensible Toolkit for Scientific Computation) [38] and extends it with all the functionality necessary for the solution of eigenvalue problems. It is written in C++ but a Fortran interface is also available. It can be downloaded from www.grycap.upv.es/slepc/ and many documents can be found in this web page.

SLEPc can be used for the solution of eigenvalue problems given in either standard or generalized form, both Hermitian and non-Hermitian, as well as other related problems such as the singular value decomposition [37]. It focuses on sparse problems. Therefore it is based on projection methods such as Krylov-Schur, Lanczos, Arnoldi, Subspace Iteration and Power/RQI. It also gives the opportunity to apply for different types of problems and spectral transformations. More information can be found in [34].

The main objects provided by SLEPc are the Eigenvalue Problem Solver (EPS) and the Spectral Transformation (ST). ST is used to compute the internal eigenvalues and accelerate the convergence. EPS is used to specify an eigenvalue problem and provides an efficient access

Problem Type	Command Line Key
Hermitian	$-eps\_hermitian$
Non-Hermitian	$-eps\_non\_hermitian$
Generalized Hermitian	$-eps\_gen\_hermitian$
Generalized Non-Hermitian	$-eps\_gen\_non\_hermitian$
GNH with positive (semi-) definite B	$-eps\_pos\_gen\_non\_hermitian$

Table 4.5: Promlem Types in SLEPc

to all eigensolvers included in the package. Furthermore, it is used to change default values of parameters such as eigenvalue number, tolerance, maximum number of iterations, etc. SLEPc is able to cope with many types of problems. Currently supported problem types are given in Table 4.5. The default problem type is non-Hermitian. So it is not necessary to change the problem type for non symmetric problems.

The available methods for solving the eigenvalue problems on SLEPc are power iteration with deflation, Arnoldi method with explicit restarting, Lanczos method, Krylov-Schur. The default method is Krylov- Schur. Each method is designed to compute the largest eigenvalues. However, it is possible to compute the smallest eigenvalues, smallest/largest real part or smallest/largest imaginary part of the eigenvalues. It is also possible to compute the interior eigenvalues with harmonic extraction or spectral transformations. In harmonic extraction the user can compute eigenvalues around a target value κ. The spectral transformation methods in SLEPc are given in Table 4.6.

Sorting criterion	options name
shift of origin	shift
spectrum folding	fold
shift-and-invert	sinvert
cayley	cayley

Table 4.6: Spectral Transformation Methods in SLEPc

In the shift of origin the matrix is shifted with the given σ value. Spectrum folding means shifting then taking the square of the matrix. By Cayley transformation a shift and an anti shift is applied to the problem. the default value for the shift and anti shift in Cayley transformations is equal.

All methods are designed to compute only one eigenvalue. It is possible to change this num-

ber, however, the user should be careful while changing the desired number of eigenvalues. As the number of eigenvalues (nev) affects the dimension of the working space (ncv), there must be a relation between nev and ncv such that they satisfy $ncv = max(2nev, nev + 15)$. This is reasonable when the small portion of spectrum is needed. However, when the large number of eigenvalues are required, changing the ncv according to this relation causes storage problems and high computational costs. So, instead of ncv another parameter, mpd, is used to handle with these problems. This parameter bound the size of the problem [37]. There is no specified relation between mpd and other parameters.

The errors are controlled by the residual vector $r = A\tilde{x} - \tilde{\lambda}\tilde{x}$. In the case of the Hermitian problems, the 2-norm is used as a bound for the absolute error in the eigenvalue. However, in the case of the non-Hermitian problems, the situation is worse as there is no simple relation similar to the case of Hermitian problems. This means that the error bounds may still give an indication of the actual error but the user should be aware that they sometimes may be wrong, especially in the case of highly non-normal matrices. The default value of the error estimate bound, *tol*, is $10e-07$. It can be changed depending on the problem.

The two objects of SLEPc, EPS and ST, are used to make the necessary changes on default values. These objects contain functions to apply the changes in the written commands. It is also possible to apply changes during the run time from the command line. An example command with descriptions is given below:

$$./ex4 \; -file\ matrix \; -eps\_nev\ 20 \; -eps\_ncv\ 40 \; -eps\_hermitian$$
$$-eps\_type\ arnoldi \; -eps\_tol\ 10e-8 \; -st\_type\ cayley \; -st\_shift\ 1.$$

This command computes 50 eigenvalues around 1 of the Hermitian matrix with tolerance $10e-8$ by Arnoldi method applied with Cayley transformations. The command at the beginning of the command line $./ex4$ represents one of the examples in SLEPc. This example takes a matrix from an exterior address and then finds its eigenvalues.

$$./ex4 \; -file\ matrix \; -eps\_nev\ 50 \; -eps\_ncv\ 100$$
$$-eps\_type\ krylovschur \; -eps\_harmonic \; -eps\_target\ 1.$$

This command says the package to find 50 eigenvalues around 1 of the non Hermitian matrix by Krylov Schur method with harmonic extraction.

$$./ex4 \; -file\ matrix \; -eps\_nev\ 10000 \; -eps\_mpd\ 600 \; -eps\_hermitian$$
$$-eps\_type \; -eps\_arnoldi \; -eps\_smallest\_magnitude.$$

This command computes 10000 smallest eigenvalues of the Hermitian matrix by Arnoldi method. It should be noticed that the parameter *mpd* is used instead of *ncv* as the number of desired eigenvalues is quite high.

SLEPc also gives an opportunity to display the error estimates during the execution of the algorithm and plotting the computed approximations of the eigenvalues at the end of the process for any problem. The reference [37] can be seen for closer details.

CHAPTER 5

NUMERICAL RESULTS

5.1 Performance of Eigensolvers

In this section, we give numerical results for computation of eigenvalues of some directed graphs. A direct comparison between the solvers described in Chapter 3 is made in terms of the accuracy. Although the scope of this study would be on non symmetric matrices, the solvers *speig* and *jdqr* and library SLEPc can also be applied to symmetric matrices. Also, the solver *jdqr* is designed to solve the problems with preconditioning which makes the solver really effective as reported in [60] and [61]. However, through this chapter, the solver *jdqr* is used without preconditioning.

The solvers *speig*, *ahbeigs* and *jdqr* are written in MATLAB while SLEPc is written in C++. Therefore, a comparison of CPU-times between these solvers is not possible.

All eigensolvers investigated in this study are designed to compute some part of the spectrum: exterior or interior eigenvalues of the spectrum. However, we try to compute as many eigenvalues as possible. For this reason default values of some parameters are rearranged to compute all spectrum. Nevertheless, we were unable to compute the whole spectrum for some eigensolvers. For instance, *speig* is designed to compute at most $n - 3$ eigenvalues where n is the size of the matrix. If n eigenvalues of the matrix are desired to compute, an error occurred and eigenvalues are not computed. On the other hand, when $n - 3$ eigenvalues are desired, it runs with a warning about the number of requested eigenvalues are high and convergence may not be reached. The maximum number of eigenvalues computed by *ahbeigs* changes depending on n and the internal relations between *nbls*, *blsz* and *maxdpol*.

All computations for MATLAB are done with version 2007a and carried out on Intel Core

2 Duo CPU with 2.1 GHz and 2 GB RAM, Windows Vista operating system. The $C++$ computations are carried out on Fedora Core 12.

5.2 Spectra of Paley Digraphs

The normalized Laplacian matrix for two Paley digraphs are used in this example. Paley digraphs are a class of strongly regular digraphs. The normalized Laplacian matrix of Paley digraph has three distinct eigenvalues. The multiplicity of the 0 eigenvalue is 1 and the other eigenvalues have multiplicities $\frac{(n-1)}{2}$ where n is the size of the matrix. The y-axis in Figures 5.1, 5.2, 5.3, 5.4, 5.5, represents the absolute values of errors between exact and approximate eigenvalues.

Figures 5.1, 5.2, 5.3, and 5.4 (a) are for $n = 103$. The number of nonzero elements of matrix is 5253 which is nearly half of the total elements which indicates that the normalized Laplacian matrix is dense. In spite of the design of algorithms, we force them to compute as many eigenvalues as possible as mentioned above. *speig* computed 100 eigenvalues and *ahbeigs* computed 96 eigenvalues with parameters *nbls*= 53 and *blsz*= 2. The other solver can compute all eigenvalues.

Figure 5.1 represents the errors for eigenvalues computed with Krylov Schur and Arnoldi methods by SLEPc and they have similar error distribution. On the other hand, the error distribution of the solver *ahbiegs* is totally different as it implements the block version (see Figure 5.3). However, despite the differences in the shape of error distribution, errors for each solver fluctuates around 10^{-15} and 10^{-17}.

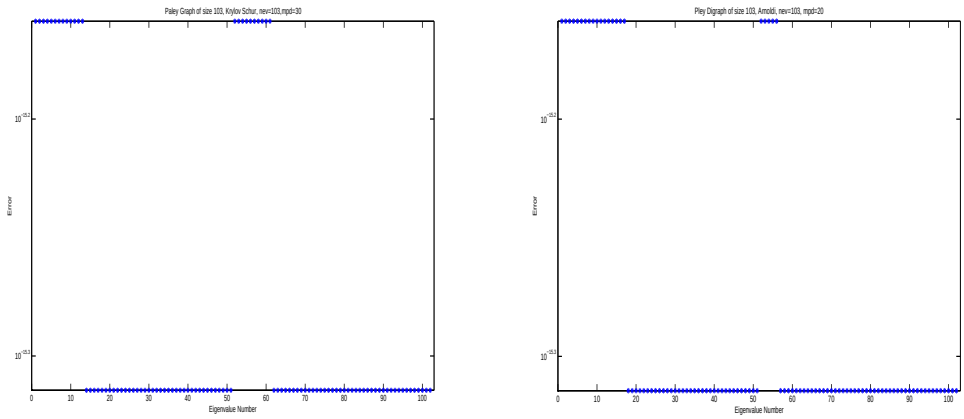
Figures 5.4 (b), 5.5, and 5.6 are prepared for $n = 1019$. *ahbeigs* calculated 1010 eigenvalues with initial parameters $nbls = 509$ and $blsz = 2$ and *speig* calculated 1016 eigenvalues whereas the other solver *jdqr* computes all eigenvalues without any problem as the Laplacian matrix of Paley digraph is diagonally dominant.

In Table 5.1 CPU times of solvers for two Paley digraphs are given to obtain a rough idea about eigenvalue computing times of solvers. These results show CPU times of solvers for computing all eigenvalues of Paley digraphs. The solver *ahbeigs* seems the fastest one as it uses block Arnoldi method. The comparison is only done between the solvers which are

Table 5.1: CPU times of Packages

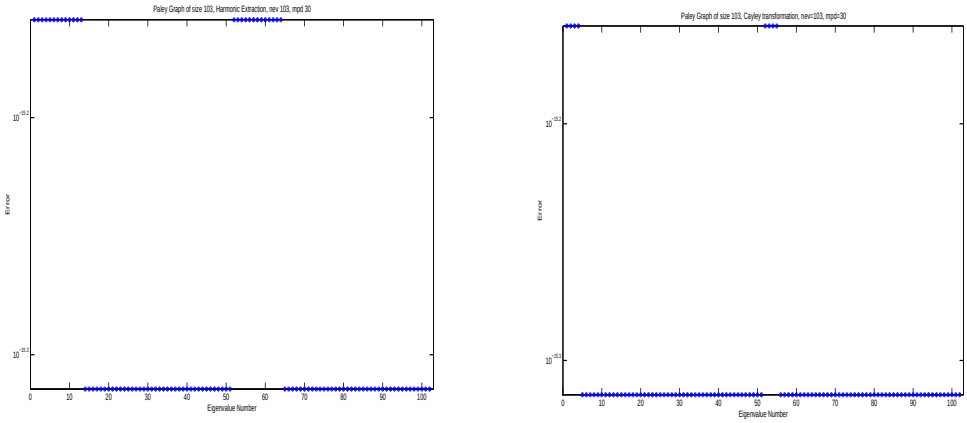
	CPU Times in Seconds	
	n=103	n=1019
speig	0.22	231.60
ahbeigs	0.30	82.38
jdqr	0.74	332.54

written in MATLAB. SLEPc is not included, as it is written in C++.



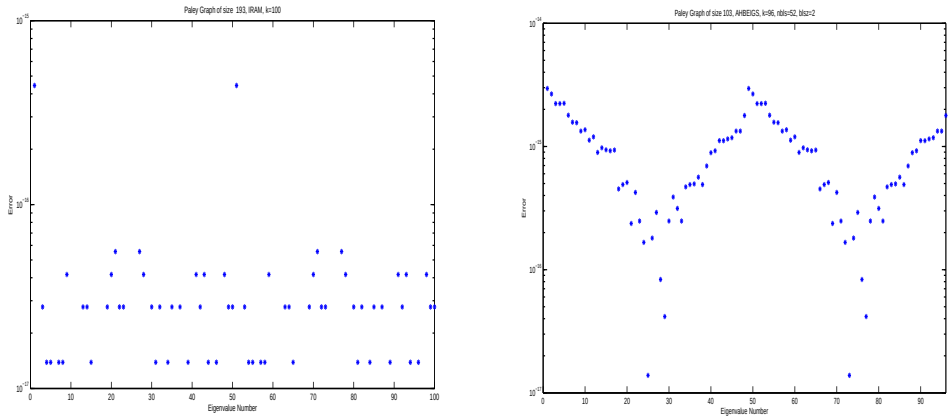
(a) (b)

Figure 5.1: Approximate eigenvalues of Paley digraph for $n = 103$. (a) SLEPc-Krylov Schur method (b) SLEPc-Arnoldi method



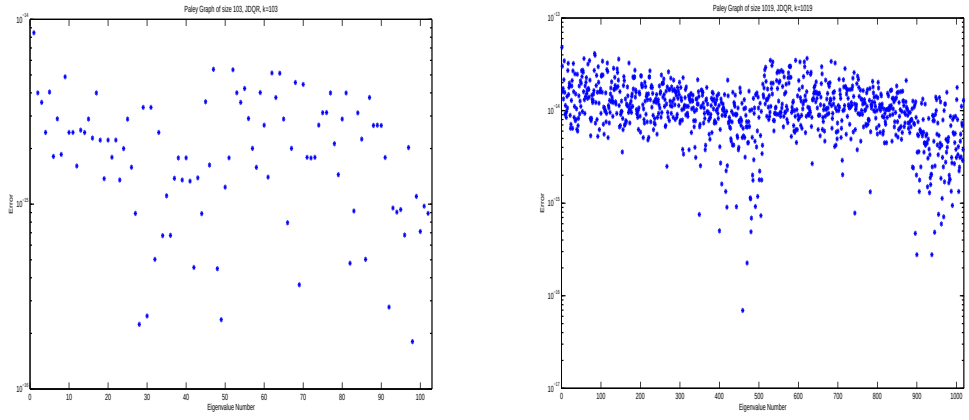
(a) (b)

Figure 5.2: Approximate eigenvalues of Paley digraph for $n = 103$. (a) SLEPc-Krylov Schur method with Cayley transformations (b) SLEPc-Krylov Schur method with harmonic extraction



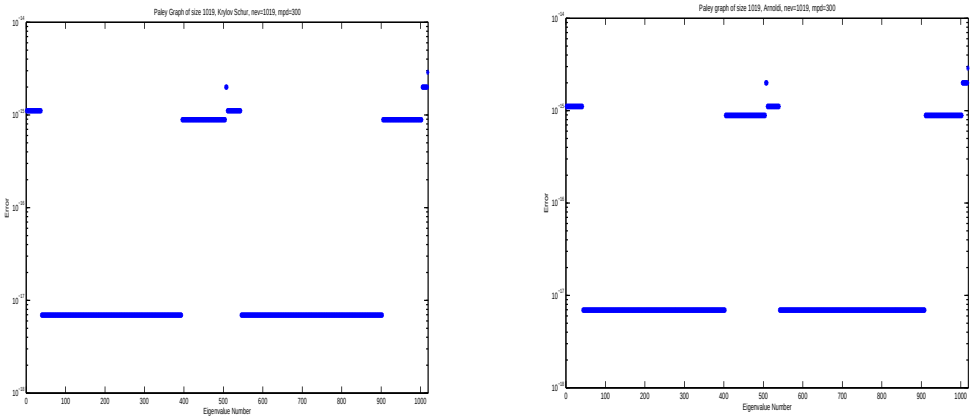
(a) (b)

Figure 5.3: Approximate eigenvalues of Paley digraph for $n = 103$. (a) speig (b) ahbeigs



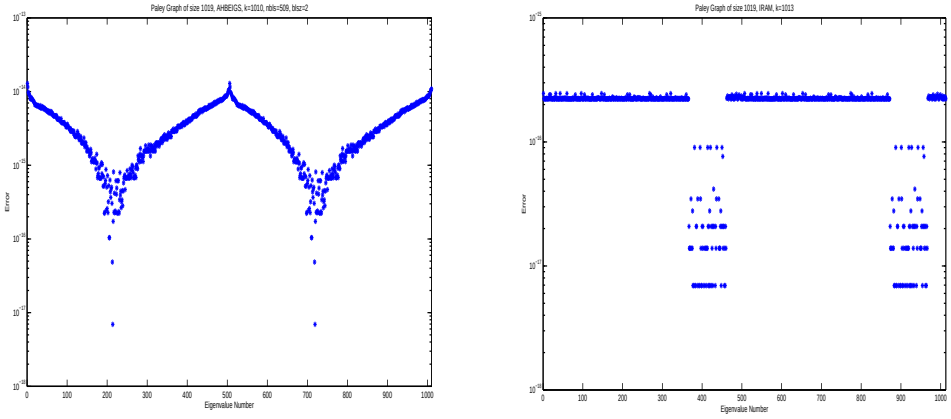
(a) (b)

Figure 5.4: Approximate eigenvalues of Paley digraph calculated by JDQR. (a) $n = 103$ (b) $n = 1019$



(a) (b)

Figure 5.5: Approximate eigenvalues of Paley digraph for $n = 1019$. (a) SLEPc-Krylov Schur method (b) SLEPc Arnoldi Method



(a) (b)

Figure 5.6: Approximate eigenvalues of Paley digraph for $n = 1019$. (a) ahbeigs (b) speig

5.3 Spectra of Empirical Networks

For the rest of this chapter, all examples are taken from the University of Florida Sparse Matrix Collection [22] that each of them represents network and belongs to Pajek group. All matrices in this collection available in adjacency matrix form but we have computed the normalized Laplacian form via related formulas given in Chapter 2.

The normalized Laplacian matrix of a network in size 396 is used as first example. It repre-

sents a citation network. As there is no available information about the exact eigenvalues of the the normalized Laplacian matrix of this network, we have computed relative residuals for eigenvalues.

The relative residual for eigensolvers is computed according to the following formula

$$r_\lambda = \frac{||Ax - \lambda x||}{||\lambda x||}, \tag{5.1}$$

where λ is the approximate eigenvalue and x is the corresponding eigenvector of it. SLEPc internally computes this value during computations and at the end of the process displays it on the screen. For other solvers we have computed the residuals according to 5.1. If $\lambda = 0$, Eq. 5.1 is undefined. Therefore, we use

$$r_\lambda = \frac{||Ax||}{x}. \tag{5.2}$$

In Figure 5.7 and 5.8 the relative residuals are calculated with respect to the eigenvalue numbers for citation network in size 396. The eigenvalue numbers are increaing from left to right and they are arranged from largest eigenvalue in magnitude to smallest one. The residual for Krylov-Schur method clusters around 10^{-14} and 10^{-15} for whole spectrum. On the other hand, the relative residual for *speig* is calculated for 393 eigenvalue because of its design as it is mentioned above. It fluctuates around 10^{-10} and 10^{-15}. The solver *ahbeigs* computed 388 eigenvalues with parameters *nbls*= 197 and *bls*= 2. The relative residuals for it clusters around 10^{-15}.

As next example we compute the normalized Laplacian matrix of a citation network in size 1059. Computations are done with SLEPc-Krylov Schur, *speig*, and *ahbeigs*. The solver *jdqr* could not work to compute all eigenvalues since the normalized Laplacian matrix is not diagonally dominant. Figures 5.7 and 5.10 show relative residuals for this citation network. *speig* calculates the 1056 eigenvalues and *ahbeigs* calculates 1050 eigenvalues with parameter values *nbls*= 529 and *blsz*= 2. The relative residuals for SLEPc Krylov-Schur and *ahbeigs* clusters around 10^{-14} and 10^{-15}. For *speig* the relative residuals fluctuates between 10^{-9} and 10^{-15}.

As next example we have computed the spectrum of a larger normalized Laplacian matrix of a citation network in size 4470. The relative residual is calculated by SLEPc-Krylov Schur method as the other solvers failed to compute all eigenvalues. They all returned errors about memory. Figure 5.14 shows the residual plot calculated by SLEPc-Krylov Schur method. According to this figure, the relative residuals for SLEPc Krylov-Schur fluctuates between 10^{-14} and 10^{-15}.

In Figures 5.1 to 5.11, the accuracy of solvers for different networks is illustrated. The accuracy of all eigenvalue solvers studied here is around machine epsilon. However, they have some pros and cons. For instance, *speig* and *ahbeigs* can not compute the all eigenvalues. Also, when size of the network increase they can not compute any of the eigenvalues. According to Figures 5.8 (a) and 5.10 (a) the solver *speig* is not good at computing interior eigenvalues when the accuracy is compared with exterior eigenvalues. For the solver *ahbeigs*, in Figures 5.8 (b) and 5.10 (b) there is not such a exact difference between exterior and interior eigenvalues. It accuracy clusters around 10^{-15} for both exterior and interior eigenvalues. *jdqr* is designed for computing eigenvalues of diagonally dominant matrices. So, it failed except for Paley digraphs. We do not mean *jdqr* can not compute any of the eigenvalues. It compute few eigenvalues. For computing many eigenvalues it needs a lot of iterations. When compared to other solvers, SLEPc-Krylov Schur seems the most efficient solver for now as it works for all network types in this study. Also, its accuracy generally fluctuates around 10^{-14} and 10^{-15}.

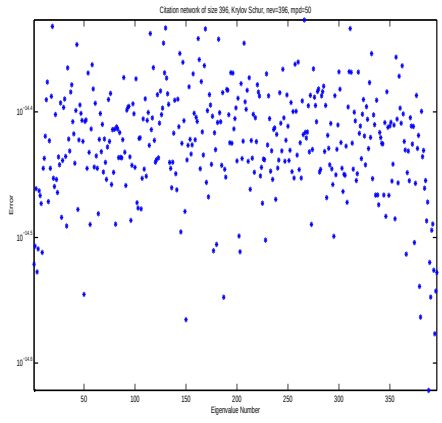
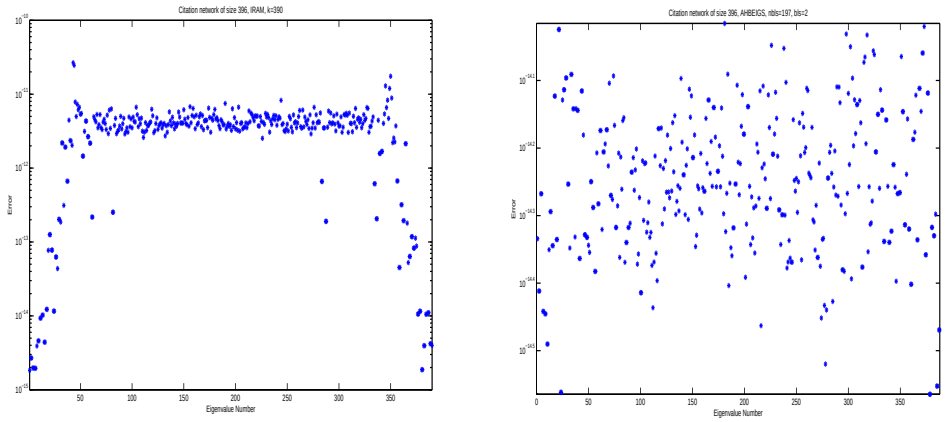


Figure 5.7: Approximate eigenvalues of citation network for $n = 396$ are calculated by SLEPc-Krylov Schur method.



(a) (b)

Figure 5.8: Approximate eigenvalues of citation network for $n = 396$. (a) speig (b) ahbeigs

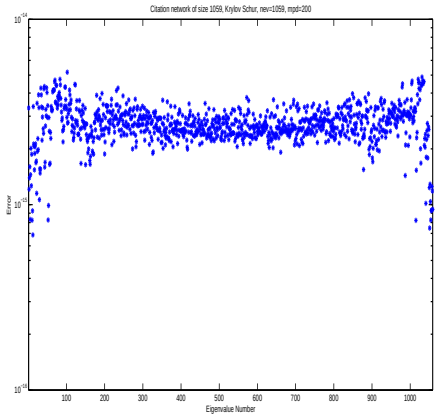
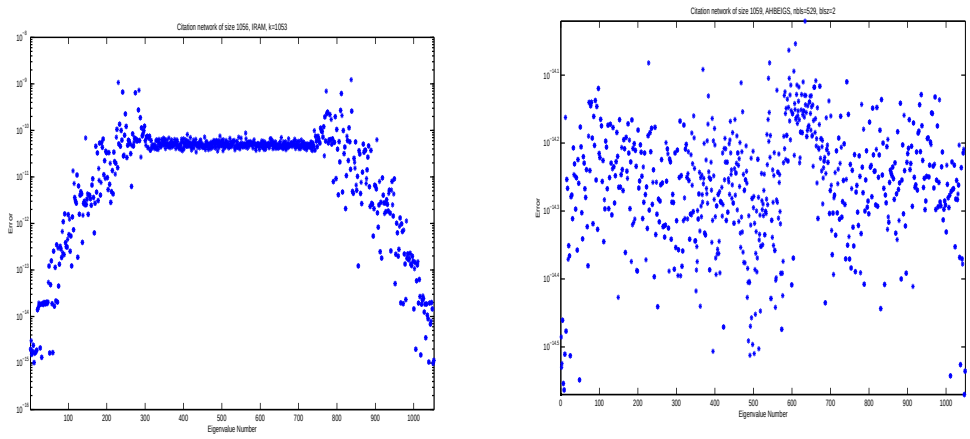


Figure 5.9: Approximate eigenvalues of citation network for $n = 1059$ are calculated by SLEPc-Krylov Schur method.



(a) (b)

Figure 5.10: Approximate eigenvalues of citation network for $n = 1059$. (a) speig (b) ahbeigs

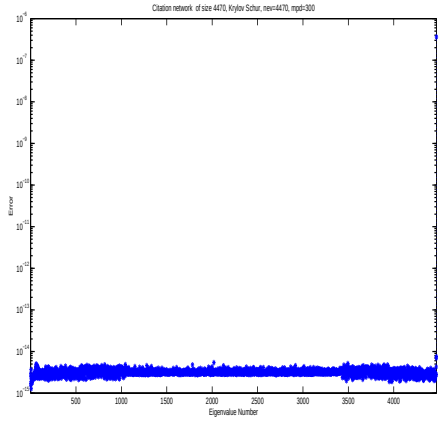


Figure 5.11: Approximate eigenvalues of citation network for $n = 4470$ are calculated by Krylov-Schur method.

5.4 Spectral Density Plots

In this section we will investigate the eigenvalue distribution of networks from the previous Section. We also provide the spectral density plots for the real part of the eigenvalues of the normalized Laplacian matrix. When the directions of all edges of a directed graph are ignored, then the spectral density of the resulting undirected graph is the same in the first-order approximation as the density of the real part of the eigenvalues of the directed graph. The distribution of the real part of the eigenvalues reflects the undirected topology of the graph

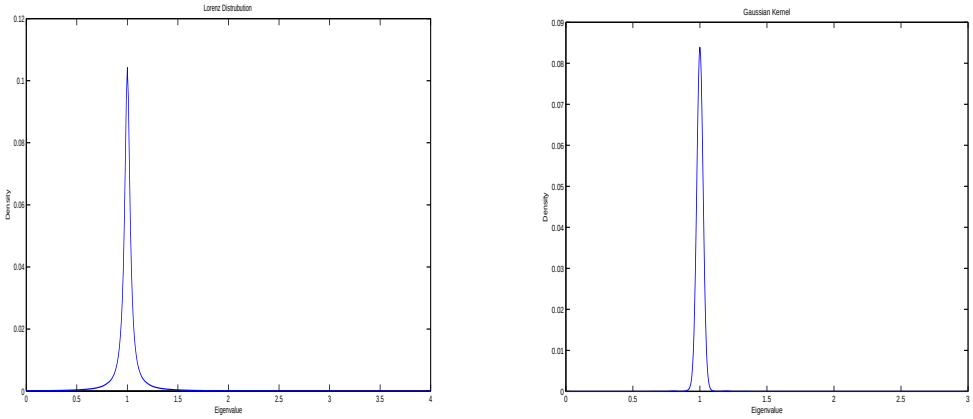
and the distribution of the imaginary part of the eigenvalues approximately gives the number of directed edges [50]. There are different plotting methods for looking into spectral density suggested in [5, 41]. The density functions with Gaussian and Lorenz kernels are used:

$$f(x) = \int g(x, \lambda) \sum_k \delta(\lambda, \lambda_k) d\lambda = \sum_k g(x, \lambda_k) \tag{5.3}$$

where $g(x, \lambda) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(x-\lambda_x)^2}{2\sigma^2})$, the Gaussian kernel or $g(x, \lambda) = \frac{1}{\pi} \frac{\gamma}{(x-m)^2+\gamma^2}$, the Lorenz distribution. Through this study, Lorenz distribution is used and as the smaller values are taken for the parameter, the finer details are emphasized.

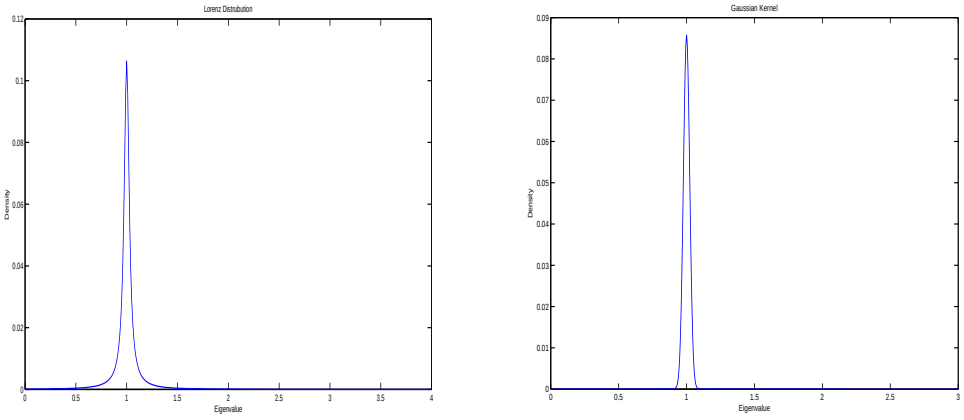
The spectral density plots of some citation networks are plotted with respect to the formulation of Lorentz and Gaussian distribution. In citation network, vertices represent a published article and directed edges stand for reference from one article to another article.

Figures 5.12 and 5.13 show the spectral density plots for real part of eigenvalues of a digraph (network) and for its underlying graph in sizes 1059 and 4470. From these figures it is seen that they show similar distributions. In all figures there is a high peak at 1 which shows that the underlying graph (undirected graph) consists of vertex doubling. Also, the peak at 0 is an evidence for the number of connected components in undirected graph (see Chapter 2). In Figure 5.14 eigenvalue distribution shows a ring structure for all networks. By Figure 5.15, a 3D visualization is provided for the same citation networks which shows symmetric eigenvalue distribution at 1. This 3D visualization is provided with Lorentz distribution suggested in [41].



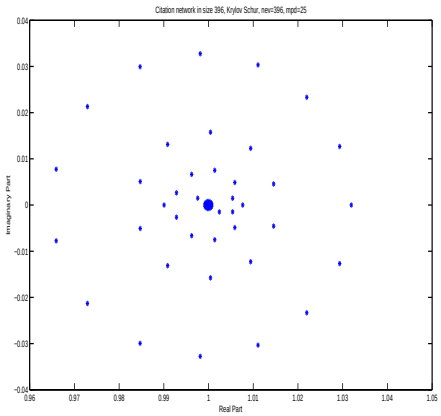
(a) (b)

Figure 5.12: The spectral density for real part of the eigenvalues of citation network for $n = 1059$ with (a) Lorenz (b) Gaussian distribution.

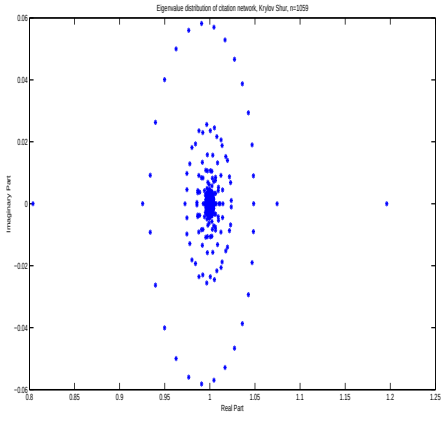


(a) (b)

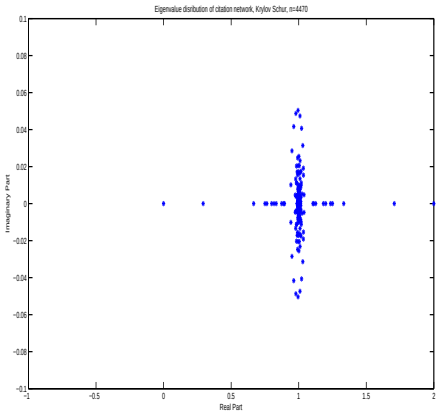
Figure 5.13: The spectral density for real part of the eigenvalues of citation network for $n = 4470$ with (a) Lorenz (b) Gaussian distribution.



(a)

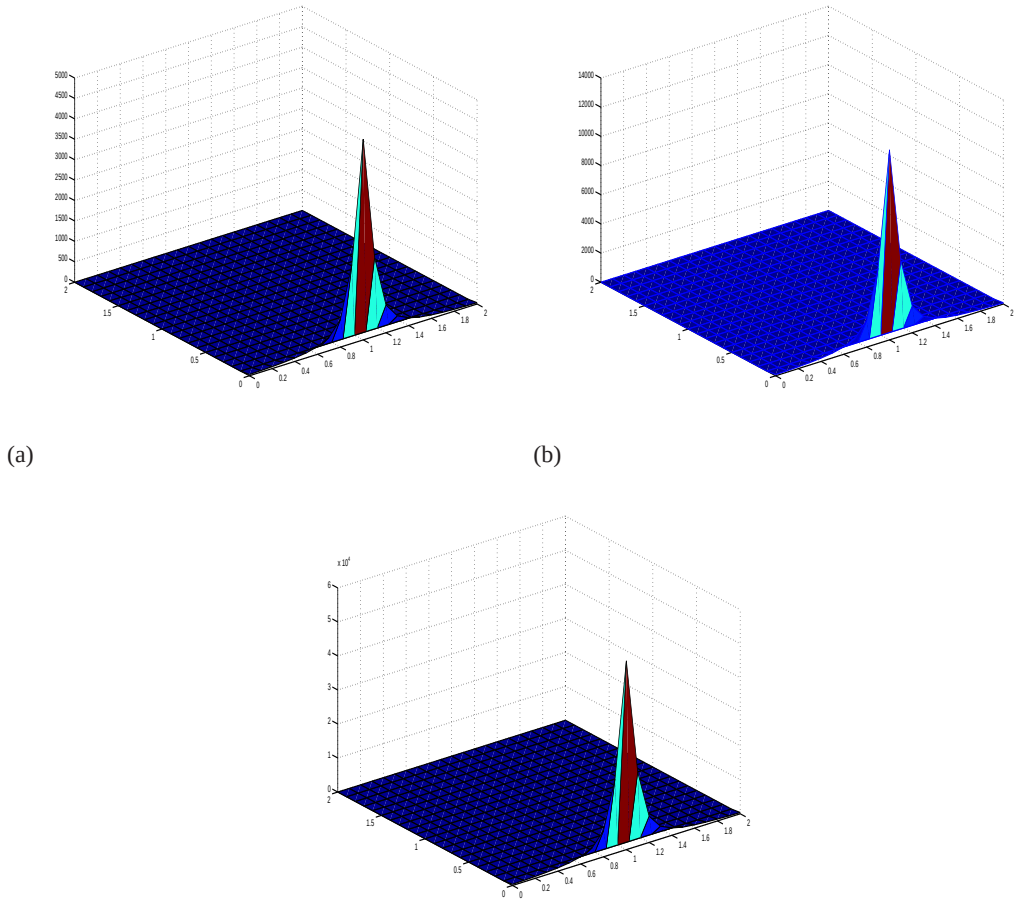


(b)



(c)

Figure 5.14: (a) Eigenvalue distribution of citation network for $n = 396$. (b) Eigenvalue distribution of citation network for $n = 1059$. (b) Eigenvalue distribution of citation network for $n = 4470$.



(a) (b)

(c)

Figure 5.15: The spectral density with Lorenz distribution for citation networks. (a) $n = 396$ (b) $n = 1059$ (c) $n = 4470$

5.5 Conclusions

The contribution of this study is on computing and analyzing the spectrum of normalized Laplacian matrices of directed graphs. We provide the comparison of the sparse eigenvalue solvers *speig*, *ahbeigs*, *jdqr*, and Krylov-Schur in SLEPc. In this comparison firstly, we use Paley digraphs as they have three distinct exact eigenvalues. Then, citation networks in different sizes are used. Both type of networks give an idea about the accuracy of eigenvalue solvers. While the absolute error is used for Paley digraphs, relative error is computed for citation networks as we do not know the exact eigenvalues of them. All the eigenvalue solvers have a good accuracy. The accuracy of each of them fluctuates between 10^{-14} and 10^{-17}.

However, the eigenvalue solvers *speig*, *ahbeigs* do not allow us to compute all eigenvalues because of their design. Also, they do not work as the size of network increases because of the memory limitations. The solver *jdqr* only works with Paley digraphs as Jacobi-Davidson methods are designed for diagonally dominant matrices. As citation networks are not diagonally dominant, it can not compute all eigenvalues. It can only compute a few eigenvalues for citation networks, however, to achieve this large number of iterations is needed. The Krylov-Schur method in SLEPc works very well with all examples in this study.

By means of the numerical examples for finding the most appropriate algorithms, we compute the all eigenvalues of some citation networks in different sizes. The eigenvalue distribution of these networks show a circular distribution. In terms of a network, this means that there is flow in networks [40, 50]. In three dimensional visualization of networks all eigenvalues gather around 1 and shows a symmetric distribution around 1. For a citation network, this means that there multiple edges for a pair of node and these edges are in opposite directions. Furthermore, spectral density plots for the real part of the eigenvalues and for the eigenvalues of undirected graph which underlies these citation networks show approximately the same eigenvalue distribution. Then, instead of finding the eigenvalue distribution of underlying graphs of digraphs, the eigenvalue distribution of the real part of eigenvalues of digraphs can be used.

Computing the whole spectrum of large matrices obtained from network applications is really challenging as it requires memory efficient algorithms. The efficient algorithm in this study, SLEPc, may fail because of memory limitations when it is applied to larger networks. For this reason, parallel computing can be applied with SLEPc algorithms or any other eigenvalues as a future work in the field of eigenvalue computation of large sparse matrices.

Spectral analysis of digraphs is a very active research area as it is thought as a special case of graphs for long times. By means of memory efficient algorithms this research area can also be extended. The relations between some exact eigenvalue of digraphs and construction type of the graph may be obtained similar to the case in protein protein interaction network. Many digraph parameters can be defined which are undefined now. By means of intense study in this area, many questions for digraphs which have not been solved yet could be answered.

REFERENCES

[1] J. Baglama, D. Calvetti, L. Reichel and A. Ruttan. Computation of a few small eigenvalues of a large matrix with applications to liquid crystal modeling, *J. Comput. Phys.*, 146:203–226, 1998.

[2] J. Baglama, D. Calvetti and L. Reichel. IRBL: an implicitly restarted block Lanczos method for large-scale Hermitian eigenproblems, *J. Sci. Comput.*, 24:1650–1677, 2003.

[3] J. Baglama. Augmented block Householder Arnoldi method. *Linear Algebra Appl.*, 429:2315–2334, 2008.

[4] J. Baglama and L. Reichel. Augmented implicitly restarted Lanczos bidiagonalization methods, *J. Sci. Comput.*, 27:19–42, 2005.

[5] A. Banerjee. *The spectrum of the graph Laplacian as a tool for analyzing structure and evolution of networks*, PhD Thesis, Fakultat fur Mathematik und Informatik der Universitat Leipzig, 2008.

[6] A. Banerjee and J. Jost. The spectral characterization of network structures and dynamics, In N. Ganguly, A. Deutsch, and A. Mukherjee, editors, *Dynamics On and Of Complex Networks*, pages 117–132. Birkhauser Boston, 2009.

[7] A. Banerjee and J. Jost. *Laplacian spectrum and protein–protein interaction networks*, Technical report, 2008, Available at arXiv.org:0705.3373v1.

[8] A. Banerjee and J. Jost. Graph spectra as a systematic tool in computational biology, *Discrete Appl. Math.*, 157:2425–2431, 2009.

[9] A. Banerjee and J. Jost. Spectral plots and the representation and interpretation of biological data, *Theory in Biosciences*, 126:15, 2007.

[10] A. L. Barabasi and R. Albert. Emergence of scaling in random networks, *Science*, 286:509–512, 1999.

[11] U. Brandes and T. Erlebach, editors. *Network Analysis: Methodological Foundations*, Lecture Notes in Computer Science, Springer, 2005.

[12] R.A. Brualdi. Spectra of Digraphs. *Linear Algebra Appl.*, 432:2181–2213, 2010.

[13] R.Brualdi and H.J. Ryser. *Combinatorial Matrix Theory*, Cambrige University Press, 1991.

[14] J. Cheeger. *A lower bound for the smallest eigenvalue of the Laplacian in Problems in analysis*, Papers dedicated to Salomon Bochner, Princeton Univ. Press, Princeton, 195–199, 1970

[15] D .Chaniotis and M.A Pai. A New Preconditioning Technique For The GMRES Algorithm in Power Flow and P-V Curve Calculations, *Electrical Power and Energy Systems*, 25:239–245, 2003.

[16] F. Chung. *Spectral Graph Theory (Cbms Regional Conference Series in Mathematics)*, CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[17] F. Chung and L. Lu *Complex Graphs and Networks*, CBMS Regional Conference Series in Mathematics. American Mathematical Society, 2006.

[18] F. Chung, L. Lu and V. Vu, Spectra of random graphs with given expected degrees, *proc Natl Acad Sci USA*, 100:6313–6318, 2003.

[19] F. Chung. The diameter and Laplacian eigenvalues of directed graphs, *Electron. J. Combin.*, 13:6, 2006.

[20] F.Chung. Laplacians and the Cheeger inequality for directed graphs, *Ann. Comb.*, 9:1–19, 2005.

[21] J.K. Cullum and R.A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1*, SIAM, 2002.

[22] T.A. Davis. The University of Florida Sparse Matrix Collection. http://www.cise.ufl.edu/research/sparse. Technical report, 1997.

[23] J. Demmel. *Templates for the Solution of Algebraic Eigenvalue Problems*, SIAM, 2000.

[24] R. Diestel. *Graph Theory*, Springer, 2005.

[25] P. Erdos and A. Renyi. On random graphs. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[26] P. Erdos and A. Renyi. On the evolution of random graphs, *Publ. Math. Inst. Hung. Acad. Sci*, 5:17–61, 1960.

[27] P. Erdos and A. Renyi. On the strengt of connenctedness of a random graph, *Acta. Math. Acad. Sci.*, 12:261–267, 1961.

[28] D.R. Fokkema, G.L. Sleijpen and H.A. Van der Vorst. Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils, *J. Sci. Compt.*, 20:94–125, 1998.

[29] L.R. Foulds. *Graph Theory Applications*, Springer Press, 1992.

[30] C.Godsil and G. Royle. *Algebraic Graph Theory*, volume 207, Springer, 2001.

[31] G.H. Golub and Q.Ye. An inverse free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems, *SIAM J. Sci. Comput.*, 24:312–334, 2002.

[32] J.L. Gross and J. Yellen. *Handbook of Graph Theory*, CRC Press, 2003.

[33] M.T. Heat. *Scientific Computing: An Introductory Survey*, Second Edition, McGraw-Hill, 2002.

[34] V. Hernandez, J. E. Roman, A. Tomas and V. Vidal. A survey of Software for Sparse Eigenvalue Problems. Technical Report STR-6, Universidad Politecnica de Valencia, 2007. Available at http://www.grycap.upv.es/slepc.

[35] V. Hernadez, J. E. Roman, A. Tomas and V. Vidal. Arnoldi Methods in SLEPc. Technical Report STR-7, Universidad Politecnica de Valencia, 2007, Available at http://www.grycap.upv.es/slepc.

[36] V. Hernadez, J. E. Roman, A. Tomas and V. Vidal. Krylov-Schur Methods in SLEPc. Technical Report STR-7, Universidad Politecnica de Valencia, 2007. Available at http://www.grycap.upv.es/slepc.

[37] V. Hernadez, J. E. Roman, A. Tomas and V. Vidal. SLEPc User Manual. Technical Report DSIC-II/24/02 - Revision 3.0.0, D. Sistemas Informaticos y Computacion, Universidad Politecnica de Valencia, 2009.

[38] V. Hernadez, J. E. Roman, A. Tomas and V. Vidal. *PETSc User Manual*, Available at http://www.mcs.anl.gov/petsc/petsc-as/snapshots/petsc-current/docs/manual.pdf, 2010.

[39] C.G. Jacobi. Über eine leichtes verfahren, die in der theorie der s¨acularstörungen vorkommenden gleichungen numerish aufzulösen, *Journal Reine Angew. Math.*, 30:51–94, 1846.

[40] J.B. Jensen and G. Gutin. *Digraphs Theory, Algorithms and Applications*, Springer, 2007.

[41] P.F. Kaluza, *Evolutionary engineering of complex functional networks*, PhD Thesis, Von der Fakultat II–Mathematik und naturwissenschaften der Technishen Universitat Berlin, Berlin, 2007.

[42] R.B. Lehoucq and D.C. Sorensen. Deflation Techniques for an implicitly Restarted Arnoldi Iteration., *SIAM, J. Matrix Anal. Appl.*, 17:789–821, 1996.

[43] R.B. Lehoucq and K.J. Maschhoff. *Implementation of an Implicitly Restarted Block Arnoldi Method*, 1997.

[44] R.B. Lehoucq, D.C. Sorensen and C.Yang. *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, 1998.

[45] C.B. Moler and G.W. Stewart. An algorithm for generalized matrix eigenvalue problems, *SIAM J. Numer. Anal.*, 10:241–256, 1973.

[46] R. Morgan. Computing Interior Eigenvalues of Large Matrices, *Linear Algebra Appl.*, 154/159:289–309, 1991.

[47] R.B. Morgan. On Restarting the Arnoldi Method For Large Nonsymmetric Eigenvalue Problems, *Math. Comput.*, 65:1213–1230, 1996.

[48] R. Morgan. Restarted Block GMRES with Deflation of Eigenvalues, *Applied Numerical Mathematics*, 54:222-236, 2005.

[49] J. Möller. Implementations of the implicitly restarted block Arnoldi method. Technical Report, Royal Institute of Technology, Dept. of Numerical Analysis and Computer Science, 2004.

[50] T. Murai. *Spectral Analysis of Directed Complex Networks*, M.S Thesis, Department of Physics, Graduate School of Science and Engineering, Aoyama Gakuin University, 2002.

[51] M.E.J. Newman. The Structure and Function of Complex Networks, *SIAM Review*, 45:167–256, 2003.

[52] B.N. Parlett. *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, N.J., 1980.

[53] D.J.D. Price. *Networks of Scientific Papers*, Science, 149:510–515, 1965.

[54] R.J. Radke. *A Matlab Implementation of the Implicitly Restarted Arnoldi Method for Solving Large Scale Eigenvalue Problems*, PhD Thesis, Rice University, 1996.

[55] Y. Saad. *Numerical Methods For Large Eigenvalue Problems*, Halsted Press, New York, 1992.

[56] Y. Saad. *Iterative Methods for Sparse Linear Systems*, SIAM, Second Edition, 2003.

[57] Y. Saad. Chebyshev Acceleration Techniques for Solving Nonsymmetric Eigenvalue Problems, *Math. Comp.*, 42:567–588, 1984.

[58] R. Schreiber and C. Van Loan. A Storage Efficient WY Representation for Products of Hoseholder Transformations, *SIAM J. Sci. Stat. Comput.*, 10:53–57, 1989.

[59] H. Simon. On a Class of Skew distribution functions, *Biometrika*, 42:425–440, 1955.

[60] G.L. Sleijpen and H.A. Van Der Vorst. A Jacobi-Davidson Iteration Method for Linear Eigenvalue Problems, *SIAM Rev*, 42:267–293, 2000.

[61] G.L. Sleijpen, H. A. Van Der Vorst, and D. R. Fokkema. Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils, *SIAM J. Sci. Comput*, 20:94–125, 1998.

[62] D.C. Sorensen. Implicitly Restarted Arnoldi/Lanczos Methods For Large Scale Eigenvalue Calculations, In *Parallel Numerical Algorithms(Hampton,VA,1994)*, volume 4, pages 119–165. Kluwer, 1997 ICASE, 1996.

[63] D.C. Sorensen. Numerical Methods For Large Eigenvalue Problems, *Acta Numer.*, 50(1):519–584, 2002.

[64] D.C. Sorensen. Implicit Application of Polynomial Filters in a k-Step Arnoldi Method, *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[65] G.W. Stewart. *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[66] G.W. Stewart. *Matrix Algorithms, Vol I*, SIAM, 1998.

[67] G.W. Stewart. *Matrix Algorithms, Vol II*, SIAM, 2001.

[68] G.W. Stewart. A Krylov-Schur Algorithm For Large Eigenproblems, *SIAM J. Matrix Anal.Appl.*, 23:601–614, 2001.

[69] H. Walker. Implementation of the GMRES Method Using Householder Transformations, *SIAM J. Sci.Compt.*, 9:152–163, 1988.

[70] D.S. Watkins. A Refined Iterative Algorithm Based on the Block Arnoldi Process for Large Unsymmetric Eigenproblems, *Linear Algebra and Its Applications*, 1998.

[71] D.J. Watts and S.H. Strogatz. Collective dynamics of ‘small-world’ networks, *Nature*, 393:440–442, 1998.

[72] K. Wu and H. Simon. Thick-Restart Lanczos Method for Large Symmetric Eigenvalue Problems, *SIAM, J. Matrix Anal. Appl.*, 22:602–616, 2001.

[73] C. Vasudev. *Graph Theory with Applications*, New Age International Publisher, 2006.

[74] Y. Zhou and Y. Saad. *Block Krylov-Schur method for large symmetric eigenvalue problems*, Numerical Algorithms, 47:341–359, 2008.