IMPROVING FEAST FOR REAL SYMMETRIC STANDARD EIGENVALUE PROBLEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KADİR ÖZÇOBAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2024

Approval of the thesis:

IMPROVING FEAST FOR REAL SYMMETRIC STANDARD EIGENVALUE PROBLEMS

submitted by **KADİR ÖZÇOBAN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Murat Manguoğlu
Supervisor, **Computer Engineering, METU** _____

Assoc. Prof. Dr. Emrullah Fatih Yetkin
Co-supervisor, **M.I.S Department, Kadir Has University** _____

Examining Committee Members:

Prof. Dr. Ahmet Coşar
Computer Engineering, Ankara Medipol Üniversitesi _____

Prof. Dr. Murat Manguoğlu
Computer Engineering, METU _____

Prof. Dr. Ahmet Oğuz Akyüz
Computer Engineering, METU _____

Date:03.09.2024

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, Surname: KADİR ÖZÇOBAN

Signature :

ABSTRACT

IMPROVING FEAST FOR REAL SYMMETRIC STANDARD EIGENVALUE PROBLEMS

ÖZÇOBAN, KADİR
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Murat Manguoğlu
Co-Supervisor: Assoc. Prof. Dr. Emrullah Fatih Yetkin

September 2024, 83 pages

Eigenvalue problems may arise from various application areas including Quantum mechanics, Computational Fluid Dynamics, Power Networks, and Machine Learning. To solve these problems, several methods for computing all or a batch of eigenvalues and eigenvectors (i.e. eigenpairs) have been proposed over the years, including both direct and iterative approaches. FEAST is a subspace-based iterative technique that simplifies the computation by projecting the problem onto a lower-dimensional subspace while preserving the eigenpairs. Despite its widespread use, FEAST is not without limitations. One of the most conspicuous problems is that a crucial matrix for the algorithm which is supposed to be orthogonal might lose its orthogonality throughout the iterations because of the well-known floating point errors or the approximate methods used to obtain it. This possible issue could cause slower convergence or even spurious eigenvalues, values that do not originally belong to the matrix. In this thesis, new as well as existing methods to improve the stability of the FEAST algorithm are investigated over various eigenvalue spectrum. Additionally, a novel method in which FEAST is preceded with inverse subspace iteration to provide better

initial guesses for the algorithm is studied. Moreover, this method is further extended in a Hybrid manner by iterating these two alternatively.

Keywords: Eigenvalue Problem, Feast, Hybrid Approaches, Sparse Matrix Computations

ÖZ

FEASTİN GERÇEK SİMETRİK STANDART ÖZDEĞER PROBLEMLERİ İÇİN İYİLEŞTİRİLMESİ

ÖZÇOBAN, KADİR
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Murat Manguoğlu
Ortak Tez Yöneticisi: Doç. Dr. Emrullah Fatih Yetkin

Eylül 2024 , 83 sayfa

Özdeğer problemleri Kuantum mekaniği, Hesaplamalı Akışkanlar Dinamiği, Güç Ağları ve Makine Öğrenimi gibi çeşitli uygulama alanlarında ortaya çıkabilir. Özdeğerlerin ve özvektörlerin (özçiftlerin) tamamını veya bir kısmını hesaplamak adına yıllar boyunca hem doğrudan hem de yinelemeli yöntemler önerilmiştir. FEAST, özçiftleri korurken sorunu daha düşük boyutlu bir alt uzaya yansıtarak hesaplamayı basitleştiren, alt uzay tabanlı yinelemeli bir tekniktir. Yaygın kullanımına rağmen FEASTin de bazı problemleri bulunmaktadır. En göze çarpan sorunlardan biri, algoritmanın çalışması için dik olması gereken önemli bir matrisin, kayan nokta hataları veya matrisi elde etmek için kullanılan yaklaşık yöntemler nedeniyle yinelemeler boyunca dikliğini kaybedebilmesidir. Bu olası sorun, daha yavaş yakınsamaya ve hatta orijinal olarak matrise ait olmayan sahte özdeğerlerin ortaya çıkmasına neden olabilir. Bu tezde, FEAST algoritmasının kararlılığını arttırmaya yönelik mevcut ve yeni yöntemler çeşitli özdeğer dağılımları üzerinde incelenmiştir. Yeni yöntem olarak algoritma için daha iyi başlangıç tahminleri sağlamak amacıyla FEASTten önce ters altuzay iteras-

yonunun koşulduğu bir kurulum üzerinde çalışılmıştır. Bu yaklaşım FEAST ve ters altuzay iterasyonunun birbirinin ardılı olacak şekilde çalıştırıldığı hibrit bir yöntem olarak genişletilmiştir.

Anahtar Kelimeler: Özdeğer Problemi, Feast, Hibrit Yöntemler, Seyrek Matris Hesaplamaları

To my beloved ones ...

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisors, Murat Manguoğlu and Emrullah Fatih Yetkin, for their unwavering support and guidance throughout my studies. Their valuable insights, constant encouragement, and mentorship have been instrumental in shaping my academic development and career path.

I am grateful to have close friends with whom, especially over the past two years, I've shared countless experiences and collected valuable, often wonderfully nonsensical memories. Specifically, I would like to thank Özge Tuğcu, Onur Tuğcu, and their lovely baby, Ferzan Tuğcu, for making me feel like a part of their family.

I feel fortunate to have Kutay Kısadere in my life, with whom I have shared a great friendship for over 15 years. Our discussions and shared experiences have been priceless, and he has always been there for me when I needed him.

I would like to thank Osman Talha Yurtoğlu and Samet Özçoban for their presence in my life, which brings me comfort. Most of this study was reviewed, analyzed, tested, and written during the days we spent together, which often ended with unhealthy meals.

Additionally, I want to express my gratitude to Alperen Canışen, who not only encouraged me to engage in sports regularly during the most stressful times of the writing process but also shared deep and often humorous conversations.

I also appreciate Seyit Emin Şanver, with whom I shared my most ridiculous and funny talks, for always cheering me up and making me feel better.

Lastly, I cannot forget to thank Umut Kolsuz, who was the first to listen to my thesis from scratch during our 10 km Anıttepe night run.

I would also like to acknowledge my other friends whose names I have not mentioned. Although we did not share anything directly related to the thesis, their presence in my

life has meant a lot to me.

Last but not least, I would like to express my deepest gratitude to my family. Without my parents, Lokman Özçoban and Özden Başak Özçoban, this thesis would not have been possible. I am grateful to them for always encouraging me to follow my dreams and for creating the circumstances necessary to achieve them. They are and will continue to be the sources of my inspiration.

My most heartfelt thanks go to my sisters, Büşra Özçoban and Berra Özçoban, for their tolerance and support as their elder brother. Their love is worth the world to me. They, along with our lovely cat, Şeker, have always brought joy to my life and will continue to do so.

I would like to acknowledge the support I received from TÜBİTAK (The Scientific and Technological Research Council of Turkey) for the project entitled "Nonlinear Dimensionality Reduction via Recycling Krylov Subspaces with Applications to Industrial Internet of Things (IIoT) Data" which was funded through the 2021 TÜBİTAK-France Bosphorus Joint Projects Call under project number 221N220.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ISI	Inverse Subspace Iteration
SISI	Shifted Inverse Subspace Iteration
Ne	Number of Quadrature Points Used on FEAST
eigNum	Number of Eigenvalues Desired to be Computed by FEAST
initDim	Constant That Multiplied by eigNum to Calculate the Dimension of the Initial Subspace Provided to FEAST

CHAPTER 1

INTRODUCTION

1.1 What are eigenvalue problems and their possible sources?

Eigenvalue problems whose further categorizations are mentioned in the following subsections have a crucial role in various fields comprising Physics, Machine Learning, Mathematics, and Statistics [1]. The mathematical representation of the problem could be given by the equation

$$Av = \lambda v, \quad v \neq 0 \tag{1.1}$$

where λ is called an eigenvalue and v is the corresponding eigenvector [2]. Accordingly, eigenvectors, which do not change their directions when the matrix is applied as an operator, correspond to the most informative directions of that matrix. For instance, this fact is used in Principal Component Analysis (PCA) as the eigenvectors of the covariance matrix show the spread and variance directions of the data [3].

As mentioned, various fields require the computation of the eigenpairs. For instance, many physical systems described by differential equations lead to eigenvalue problems, or Markov chains' steady-state behavior is analyzed using transition matrices eigenvalues. As more examples, in the Machine Learning field, eigenvalues and eigenvectors are used in algorithms such as PCA, spectral clustering, and manifold learning. Even for economics, they are utilized for input and output analysis.

As it might have been guessed, the form given in Eq.(1.1), which is called the standard eigenvalue problem, is not the only form of eigenvalue problem. In the following subsection, the types of the problems are further detailed.

1.1.1 Categorizations of the problem

The general representation of the linear eigenvalue problems, which is called the generalized eigenvalue problem[4], can be given by

$$Av = \lambda Bv, \quad v \neq 0.$$

which is denoted by the pair (A, B) that is called the pencil of the problem [5]. It could be seen that the standard eigenvalue problem is a special case of the generalized one where $B = I$. This categorization can be further extended by the non-linear eigenvalue problems, refer to [6] for the detailed analysis.

Instead of the structure of the problem, the properties of the matrices could also be used for categorizing. Firstly, whether the elements of the matrices are real or complex is a way to split the problems. Followingly, the properties of the matrices could be used.

1. **Symmetric Generalized Eigenvalue Problem**
    - **Matrix Structure:** A and B are symmetric
    - **Properties:** Eigenvalues are real, and eigenvectors corresponding to distinct eigenvalues are $B-$orthogonal
2. **Hermitian Generalized Eigenvalue Problem**
    - **Matrix Structure:** A and B are hermitian (complex symmetric)
    - **Properties:** Eigenvalues are real if B is positive definite
3. **Non-symmetric Generalized Eigenvalue Problem**
    - **Matrix Structure:** A and B are non-symmetric
    - **Properties:** Eigenvalues can be complex, and eigenvectors are not necessarily orthogonal also can be defective

A complex matrix is called positive definite if the following constraint

$$\mathfrak{R}[v^h Av] > 0, \quad \forall v \neq 0$$

which can be directly applied to real ones, holds. Note that the matrices satisfying the greater than equal to condition instead of greater than, are called the positive semi-definite matrices.

In this thesis, real symmetric standard eigenvalue problems are studied; therefore, in the following subsection further properties of them are mentioned.

1.2 Real Symmetric Eigenvalue Problem

The scope of the thesis contains the following problem:

$$Av = \lambda v$$

where A is a real symmetric matrix, having the dimension of n. Namely, in the following sections $A \in \mathbb{R}^{n \times n}$ unless otherwise stated. One of the important properties of it is that eigenvalues are real. This can be proved by the following steps.

$$\begin{aligned}
Av &= \lambda v \quad \textit{take complex conjugate of both sides, as } A \textit{ is real } \overline{A} = A \\
A\overline{v} &= \overline{\lambda}\overline{v} \quad \textit{take transpose of both sides, as } A \textit{ is symmetric } A^T = A \\
\overline{v}^T A &= \overline{\lambda}\overline{v}^T \quad \textit{multiply both sides with } v\textit{, as by assumption } v > 0 \\
\overline{v}^T Av &= \overline{\lambda}\overline{v}^T v \quad \textit{by definition } Av = \lambda v \\
\overline{v}^T \lambda v &= \overline{\lambda}\overline{v}^T v \quad \textit{divide both sides by } v,\ \textit{as } \overline{v}^T v > 0 \\
\overline{\lambda} &= \lambda \quad \textit{meaning that } \lambda \textit{ is real}
\end{aligned}$$

Another crucial property of the problem is that eigenvectors corresponding to different eigenvalues of the matrix are orthogonal to each other, which can be proved as follows.

$$
\begin{aligned}
Av_i &= \lambda_i v_i \quad \textit{by assuming } \lambda_i \neq \lambda_j \\
Av_j &= \lambda_j v_j \quad \textit{multiply the first one with } v_j^T \\
v_j^T A v_i &= v_j^T \lambda_i v_i \quad \textit{multiply the second one with } v_i^T \\
v_i^T A v_j &= v_i^T \lambda_j v_j \quad v_j^T A v_i = v_i^T A v_j \textit{ as } A \textit{ is symmetric} \\
v_j^T \lambda_i v_i &= v_i^T \lambda_j v_j \quad \textit{by assumption } \lambda_i \neq \lambda_j \\
v_j^T v_i &= 0 \quad \textit{meaning that } v_i \textit{ and } v_j \textit{ are orthogonal}
\end{aligned}
$$

As the problem has been well-known for many years, there have been various approaches to compute the eigenpairs. In the following subsection, one of the earliest is demonstrated.

1.2.1 The relation between characteristic equation and eigenpairs

A straightforward investigation of Eq.(1.1), might suggest the solution of the following linear system to compute eigenpairs

$$(A - \lambda I)v = 0. \tag{1.2}$$

The solution seems basic by letting $v = 0$; however, by definition, an eigenvector is not allowed to be a zero vector, requiring an improvement. Accordingly, further investigation shows that the null space of $(A - \lambda I)$ must have more than one vector, indicating the singularity of the matrix [7]. Therefore, by considering the relation between the determinant and null space (the determinant equals zero means that the null space contains a vector or vectors other than zero) the equation could be enhanced by taking the determinant as follows:

$$det(A - \lambda I) = 0 \tag{1.3}$$

meaning that λ values satisfying the equation, called the characteristic equation, are eigenvalues [7]. Because if for any λ Eq.(1.3) equals zero, it means there is at least one vector, called the corresponding eigenvector of λ, other than the zero vector sat-

isfying Eq.(1.2). In other words, eigenvalues are the roots of the characteristic equation given in Eq.(1.3), and eigenvectors can be computed by substituting them to Eq.(1.2). Even though this method could be used for the eigenpair computation, as the dimension of matrices increases solving the characteristic equation becomes computationally expensive; therefore, various paradigms, some of which are provided in the following section, have been proposed. Nevertheless, [8] can be referred to show the relation between characteristic equations and eigenvalues.

As mentioned, in chapter 2 various methods, one of which is FEAST, are detailed. FEAST is a subspace-based iterative eigensolver, utilizing contour integration techniques, that requires solutions to linear systems, which could compute eigenpairs lying in an interval. These features enable multiple levels of parallelism, making it favorable for parallel architectures [9]. However, FEAST as well as other solvers mentioned in the chapter are not perfect, having various deficiencies. These are also mentioned in the same chapter. Followingly, the improvements that the thesis proposes to overcome these possible problems are outlined in chapter 3. Afterward, the results of various methods over both synthetic and real-life matrices are shown in chapter 4 after which the findings of the thesis are discussed in chapter 5.

CHAPTER 2

LITERATURE REVIEW

In this chapter, various paradigms for computing eigenpairs are mentioned. Firstly, decomposition-based methods, which alter the structure of the matrix to compute all eigenpairs are given. Followingly, approaches, called subspace-based methods, that iteratively improve an initial subspace to compute a portion of eigenpairs are explained. Afterward, methods that consider eigenpair computation as a minimization problem are shown, followed by an extensive investigation of FEAST to conclude the chapter.

2.1 Decomposition Based Methods

Methods explained in this section iteratively alter the input matrix by utilizing various decompositions to transform the structure of it so that the eigenvalues are relocated at the diagonal. Consequently, the computation of eigenpairs simplifies. These methods are mostly used for computing all eigenvalues.

2.1.1 LU

In 1958, early stage of eigenpair computations, Rutishauser proposed an LR transformation (nowadays called LU decomposition) based method to obtain all eigenpairs of a matrix [10]. The method includes two distinct phases: LR transformation of a matrix and construction of a new one by the obtained triangular matrices. Initially, the input matrix $A = A_1$ is decomposed to an upper and a lower triangular matrix $A_1 = L_1 R_1$. Subsequently, a new matrix A_2 is formed by reversing the multiplica-

tion: $A_2 = R_1 L_1$. The crucial observation is that A_1 and A_2 have the same eigenvalues as $A_2 = R_1 A_1 R_1^{-1}$. Hence, A has the same eigenvalues as any A_k where A_k is the matrix obtained at the $(k-1)^{th}$ iteration of the same procedure, the pseudo-code is given in Alg.(1). Pointing that, the method utilizes the convergence of the A_k, under certain conditions, to an upper triangular matrix where eigenvalues lie on the diagonal as $k \to \infty$. Even though certain conditions that guarantee the convergence of A are not in the scope of the thesis, it is important to note that being Hermitian and positive definite is a sufficient condition. However, the method has not been preferred since the LR transformation is not guaranteed to be stable [11].

Algorithm 1 LR Transformation for Eigenvalue Computation

Inputs: A, *tolerance* ϵ, *maximum iterations max_iter*

Output: $\lambda_{1,2,\ldots,n}$

$A_0 = A$
for `k=1,2,...,max_iter` **do**
    $A_{k-1} = L_k R_k$ ▷ Perform LR decomposition
    $A_k = R_k L_k$ ▷ Form the product
    if $\|A_k - A_{k-1}\| < \epsilon$ **then** ▷ Test for convergence
        break
    end if
end for
Return $\lambda_{1,2,\ldots,n}$ (the diagonal elements of A_k)

2.1.2 QR

In 1961, Francis stated that as the LR transformation is not always stable, especially for large matrices, and the process breaks down if the pivotal element is zero, the substitution of a unitary transformation, in his case QR transformation, is preferable in terms of numerical stability [11]. Accordingly, by applying QR transformation instead of LR in the same iterative process, the algorithm was enhanced. It was also proven by Francis that if certain conditions are fulfilled, the updated matrix converges to an upper triangular. However, it is more expensive than the LR, due to the high computational cost of QR factorization. The improved version is given in Alg.(2).

However, decomposition-based methods are not preferred as generally instead of all, a batch of eigenpairs are required. Therefore, subspace-based approaches which are given in the following subsection are generally utilized as they could compute a portion of eigenpairs.

Algorithm 2 QR Factorization for Eigenvalue Computation

Inputs: A, *tolerance* ϵ, *maximum iterations max_iter*

Output: $\lambda_{1,2,\ldots,n}$

$A_0 = A$
for `k=1,2,...,max_iter` **do**
    $A_{k-1} = Q_k R_k$ ▷ Compute the QR decomposition
    $A_k = R_k Q_k$ ▷ Form the product
    if $\|A_k - A_{k-1}\| < \epsilon$ **then** ▷ Test for convergence
        break
    end if
end for
Return $\lambda_{1,2,\ldots,n}$ (the diagonal elements of A_k)

2.2 Subspace Based Methods

Contrary to methods that improve the structure of the matrix to ease the eigenvalue computation explained in 2.1, this section summarizes methods that iteratively alter a subspace to converge the eigenvectors of the matrix. The number of eigenpairs computed by these methods can at most be the dimension of the initial subspace. These methods are preferred if a part of the eigenvalue spectrum needs to be computed.

2.2.1 Single Vector Iterations

Initially, algorithms iterating over a single vector are explained. As the dimension of the vector is one, only a single eigenpair of the matrix is computed.

2.2.1.1 Power Method

One of the earliest and simplest methods for the single eigenpair computation is the power method [12]. Similar to those discussed in 2.1, the method has an iterative computational approach. However, each step only consists of a matrix-vector multiplication; consequently, only the dominant eigenvalue λ_1 (the largest one in absolute value) and the corresponding eigenvector v_1 are converged. Initially, the method starts with a random vector u_0 and computes a new one by multiplying it with A: $u_1 = Au_0$. The continuation of this operation yields $u_k = Au_{k-1}$ clearly indicating $u_k = A^k u_0$. The inspiration of the power method is that $(A^k u_0)/||A^k u_0||$ can be a good approximation for v_1 [13]. The power method is given in Alg.(3).

Algorithm 3 Power Method

Inputs: A, *initial vector* u, *tolerance* ϵ, *maximum iterations max_iter*

Output: λ_1, v_1

1: **for** `k=1,2,...,max_iter` **do**
2: $w = Au$ ▷ Compute matrix-vector product
3: $u = \frac{w}{\|w\|}$ ▷ Normalize vector
4: $\lambda = u^T Au$ ▷ Approximate eigenvalue
5: **if** $\|Au - \lambda u\| < \epsilon$ **then** ▷ Test for convergence
6: **break**
7: **end if**
8: **end for**
9: **Return** λ, v

One might notice that the normalization process is applied at each iteration contrary to the previous discussion. Nevertheless, the two techniques are equivalent since the matrix-scalar multiplication commute and the two approaches result in a unit vector [13]. It is worth noting that theoretically, the initial vector could be in the null space of A, consequently leading to a zero vector at each iteration. Nonetheless, even in this case, because of the floating point errors, the outcome would never be exactly zero; hence, the method converges as expected.

The convergence of the power method relies on the fact that any vector u can be expressed by a complete set of fundamental eigenvectors of A as $u = \alpha_1 v_1 + \alpha_2 v_2 +$

$... + \alpha_n v_n$ where $|\lambda_n| \leq ... \leq |\lambda_2| \leq |\lambda_1|$. The careful investigation of the fact would yield

$$\begin{aligned} u &= \alpha_1 v_1 + \alpha_2 v_2 + ... + \alpha_n v_n \\ A^k u &= \alpha_1 A^k v_1 + \alpha_2 A^k v_2 + ... + \alpha_n A^k v_n \\ &= \alpha_1 \lambda_1^k v_1 + \alpha_2 \lambda_2^k v_2 + ... + \alpha_n \lambda_n^k v_n \\ &= \lambda_1^k \left(\alpha_1 v_1 + \alpha_2 (\frac{\lambda_2}{\lambda_1})^k v_2 + ... + \alpha_n (\frac{\lambda_n}{\lambda_1})^k v_n \right). \end{aligned}$$

Moreover, as λ_1 is the dominant one $(\lambda_i/\lambda_1)^k$ is zero for $2 \leq i \leq n$ as $k \to \infty$ by assuming $\lambda_2 < \lambda_1$. Leading that

$$\lim_{k \to \infty} (A^k u)/||A^k u|| = (\alpha_1 \lambda_1^k v_1)/||\alpha_1 \lambda_1^k v_1||.$$

This observation about the convergence of the power method implies two properties: the method's convergence rate is dependent on $\frac{|\lambda_2|}{|\lambda_1|}$ and if the magnitude of λ_1 and λ_2 are the same the method can not converge as none can dominate the other.

2.2.1.2 Inverse Power Method

Previously, it was demonstrated that the power method is a tool for computing the dominant eigenpair of a matrix. However, many problems require the smallest eigenpair in absolute value. In such cases, the power method is not directly applicable, requiring a modification which is based on the well-known property that if A is an invertible matrix having real, non-zero eigenvalues $|\lambda_n| \leq ... \leq |\lambda_2| \leq |\lambda_1|$, then the eigenvalues of A^{-1} are $|\frac{1}{\lambda_n}| \geq ... \geq |\frac{1}{\lambda_2}| \geq |\frac{1}{\lambda_1}|$. Hence, substituting A with the inverse in the power method as $u_k = A^{-1} u_{k-1}$ is sufficient. However, if the matrix is dense the computation of the inverse requires 8/3 n^3 arithmetic operations, and the existence of it is not always guaranteed; therefore, LU factorization, which needs 2/3 n^3 arithmetic operations, is generally applied to speed up the process. Accordingly, this observation leads to Alg(4). Alternatively, an iterative method for solving the system in line 3 can also be used.

The same convergence analysis and properties of the power method can be applied to the inverse power method.

Algorithm 4 Inverse Power Iteration

Inputs: A, *initial vector* u, *tolerance* ϵ, *maximum iterations max_iter*

Output: λ_n, v_n

$A = LU$ ▷ LU decomposition for inverse computation
for `k=1,2,...,max_iter` **do**
  $w = U^{-1}L^{-1}u$ ▷ Compute matrix-vector products
  $u = \frac{w}{\|w\|}$ ▷ Normalize vector
  $\lambda = u^T Au$ ▷ Approximate eigenvalue
  if $\|Av - \lambda u\| < \epsilon$ **then** ▷ Test for convergence
    break
  end if
end for
Return λ, u

2.2.1.3 Shifted Inverse Power Iteration

In 2.2.1.1 and 2.2.1.2, methods to compute the extreme -largest and smallest- eigenpair of a matrix were introduced. However, these methods may not suffice, as many problems require finding the eigenpair closest to a scalar value. Hence a shift is required by utilizing that if λ is an eigenvalue of A, then $\frac{1}{\lambda-\sigma}$ is an eigenvalue of $(A-\sigma I)^{-1}$. Therefore, if λ is the closest eigenvalue to σ, then $\frac{1}{\lambda-\sigma}$ will be the dominant eigenvalue of the shifted matrix letting the method converge to the corresponding eigenvector. This leads to the Alg.(5) for which the same analysis can be applied.

2.2.1.4 Rayleigh Quotient Iteration

The Rayleigh quotient of a vector is given in the form

$$\sigma(u) = \frac{u^T Au}{u^T u}.$$

Based on the eigenvalue definition, one can notice that $\sigma(u) = \lambda$ is an eigenvalue if u is an eigenvector. Hence, the formula can be derived by answering the question that given an approximate vector u what is the best approximate σ to be its eigenvalue, indicating the minimization of $||Au - u\sigma||$. On the contrary to the shifted inverse

Algorithm 5 Shifted Inverse Power Iteration

Inputs: A, *initial vector* u, *tolerance* ϵ, *maximum iterations max_iter*

Output: λ_m, v_m

1: $A - \sigma I = LU$ ▷ LU decomposition for inverse computation
2: **for** `k=1,2,...,max_iter` **do**
3: $w = U^{-1}L^{-1}u$ ▷ Compute matrix-vector products
4: $u = \frac{w}{\|w\|}$ ▷ Normalize vector
5: $\lambda = u^T Au$ ▷ Approximate eigenvalue
6: **if** $\|Au - \lambda u\| < \epsilon$ **then** ▷ Test for convergence
7: **break**
8: **end if**
9: **end for**
10: **Return** λ, u

power method in which the eigenvector of the eigenvalue that is closest to the input scalar is computed, the Rayleigh quotient function computes the best scalar in terms of being the eigenvalue of the input initial guess vector. On the nature of this explanation, the iterative execution of these accelerates the convergence (quadratic convergence) of the shifted inverse power method [14]. Rayleigh Quotient Iteration is given in Alg.(6).

Indeed, the method is the same as the shifted inverse power method except the shift scalar is dynamically computed by the Rayleigh quotient function at each iteration. However, dynamic shifts result in the change of the coefficient matrix at the linear system in line 3; therefore, if LU factorization is used, the matrix has to be factorized at every iteration. Finally, it is worth noting that the method converges if the input matrix is Hermitian, refer to [15] for further analysis.

2.2.1.5 Deflation Techniques

In certain application areas, the computation of the second λ_2 (or higher) dominant eigenpair of a matrix is required. One of the most straightforward and well-known techniques is to utilize "deflation", for example, a rank one modification to the matrix

Algorithm 6 Rayleigh Quotient Iteration

Inputs: A, *initial vector* u, *tolerance* ϵ, *maximum iterations max_iter*

Output: λ_m, v_m

$\lambda = \frac{u^T A u}{u^T u}$ ▷ Approximate eigenvalue by Rayleigh-quotient
for `k=1,2,...,max_iter` **do**
    $(A - \lambda I)w = u$ ▷ Solve shifted linear system
    $u = \frac{w}{\|w\|}$ ▷ Normalize vector
    $\lambda = u^T A u$ ▷ Update eigenvalue by Rayleigh-quotient
    if $\|Au - \lambda u\| < \epsilon$ **then** ▷ Test for convergence
        break
    end if
end for
Return λ, u

so that the dominant eigenvalue λ_1 is displaced while others are kept the same [12]. One of the well-known deflation techniques is Wielandt Deflation with One Vector in which A is modified as

$$A = A - \sigma v_1 u^T$$

where σ is the appropriate shift, v_1 is the dominant eigenvector, and u is an arbitrary vector such that $v_1^T u = 1$. After the modification, the power method is applied to the altered matrix to compute the second dominant eigenpair. Furthermore, the computation of higher eigenpairs requires the iterative execution of the process.

2.2.2 Subspace Iteration Methods

In 2.2.1, various algorithms to compute an eigenpair lying on the particular position of the eigenspectrum were demonstrated. However, in most cases, more than one eigenpair is required, necessitating the improvement of the previous methods to compute other eigenpairs. Hence, in this section, methods for computing more than one eigenpair of the input matrix by iterating a subspace are explained.

2.2.2.1 Simple Subspace Iteration

One of the most elementary ideas to compute more than one dominant eigenpair of a matrix is iteratively computing the current highest and deflating it to compute others. However, this approach requires a lot of computation; hence, is not practically applicable. Nevertheless, this can be further improved by extending the dimension of the initial subspace, providing an m-dimensional matrix instead of a vector where m is the number of desired eigenpairs. Although this idea leads to the power method where the only difference is the dimension of the initial matrix, the desired eigenpairs can not be computed as in that case, all column vectors of the initial matrix converge to the eigenvector of the dominant eigenvalue. Therefore, applying QR factorization to the iterated matrix, as it orthogonalizes the columns vectors, is required to ensure convergence to all eigenvectors, leading to Alg.(7). The relation between subspace iteration and QR/LU iterations is investigated in [16].

Algorithm 7 Simple Subspace Iteration

Inputs: A, $U \in \mathbb{R}^{n \times m}$, *tolerance* ϵ, *maximum iterations max_iter*

Output: $\Lambda_{1,2,\ldots,m}$, $V_{1,2,\ldots,m}$

for `k=1,2,...,max_iter` **do**
  $W = AU$ ▷ Compute the matrix product
  $W = QR$ ▷ Perform QR decomposition
  $U = Q$ ▷ Update subspace
  $\Lambda = U^T AU$ ▷ Approximate eigenvalues
  if $\|AU - U\Lambda\| < \epsilon$ **then** ▷ Test for convergence
    break
  end if
end for
Return Λ, U

Under a few assumptions, the subspace iteration algorithm that works on a subspace of dimension m convergences at a rate of $|\lambda_{m+1}|/|\lambda_m|$, where λ_{m+1} is the eigenvalue with the $(m+1)^{th}$ largest magnitude. Moreover, one possible way to reduce the computational cost might be applying QR factorization not at each iteration but at each k whose best possible value depends on the convergence rate [12]. Furthermore,

the converged eigenvector can be deflated to reduce the cost per iteration.

As one might notice, this method computes the largest m eigenpairs; however, there might be a need for computing the smallest or interior m. Therefore, in these situations, the modification of the method in a similar way to how the (shifted) inverse power method is constructed from the power method is required. Afterward, the same convergence analysis can be applied.

2.2.2.2 Subspace Iteration with Projection

One might recall in 2.2.1.4 the Rayleigh quotient and the power method are ensembled to increase the performance. The same idea, combining two different methodologies, can be applied to the simple subspace iteration and Rayleigh-Ritz method which resembles the Rayleigh quotient extended to higher dimensions, the deeper theoretical analysis can be found in [17]. This observation leads us to Alg.(8).

Algorithm 8 Subspace Iteration with Projection

Inputs: $A, U \in \mathbb{R}^{n\times m}$, *tolerance* ϵ, *maximum iterations max_iter*

Output: $\Lambda_{1,2,\ldots,m}$, $V_{1,2,\ldots,m}$

1: **for** `k=1,2,...,max_iter` **do**
2: $W = AU$ ▷ Compute the matrix product
3: $W = QR$ ▷ Perform QR decomposition
4: $\hat{A} = Q^T AQ$ ▷ Project A onto a smaller dimensional subspace
5: $\hat{A} = V\Lambda V^T$ ▷ Compute the eigenvalue decomposition of reduced A
6: $U = QV$ ▷ Update subspace
7: **if** $\|AU - U\Lambda\| < \epsilon$ **then** ▷ Test for convergence
8: **break**
9: **end if**
10: **end for**
11: **Return** Λ, U

It is important that the projection step, line 4 of the algorithm, indeed yields a smaller problem size focusing only on the dimension of the initial provided subspace. Therefore, the computational cost of the eigenvalue problem required to be solved becomes

cheaper, especially for the problems where the number of the required eigenvalues is much smaller than the dimension of the problem, even though the reduction step causes an additional cost over the simple subspace method.

2.2.2.3 Krylov Subspaces Based Approaches

Krylov subspace methods are widely used techniques for different scientific problems, their explained construction way and usage in the section is only for eigenpair computations. For a detailed analysis of the origin and usage areas [18] is suggested.

In the power method, a sequence of vectors $A^k u$ is computed; however, the knowledge of previous vectors is discarded at each iteration as only $A^k u/||A^k u||$ vector is analyzed in terms of convergence. Therefore, one can argue that there is information loss and all previous vectors should be considered during the eigenpair computation. The critique leads to the following definition of Krylov subspace methods. The matrix

$$K^m(u) = K^m(u, A) = [u, Au, ..., A^{m-1}u]$$

generated by the vector u is called the Krlyov matrix. Its columns span the Krylov subspace

$$\mathcal{K}^m(u) = \mathcal{K}^m(u, A) = \text{span}\{u, Au, ..., A^{m-1}u\}.$$

Indeed, it is well-established from the power method that the vectors $A^k u$ converge to the dominant eigenvector of A. Hence, this basis tends to be extremely ill-conditioned and unsuitable for numerical computing, necessitating orthogonalization. The requirement can be fulfilled by applying the classical Gram-Schmidt orthogonalization process to the basis vectors in their natural order.

Assuming that $\{q_1, q_2, ..., q_i\}$ is an orthonormal basis for $\mathcal{K}^i(u)$, q_{i+1} can be computed by first orthogonalizing $A^i u$ against $q_1, q_2, ..., q_i$,

$$y_{i+1} = A^i u - \sum_{k=1}^{i} q_k q_k^T A^i u$$

and normalizing the result vector $q_{i+1} = y_{i+1}/||y_{i+1}||$. As a result, $\{q_1, q_2, ..., q_i, q_{i+1}\}$ is an orthonormal basis for $\mathcal{K}^{i+1}(u)$.

Moreover, a careful investigation of the Krylov subspace reveals that the orthogonalization process can be performed more efficiently since

$$\begin{aligned}
\mathcal{K}^{i+1}(u) &= [u, Au, A^2u, ..., A^iu] && (q_1 = u/||u||) \\
&= [q_1, Aq_1, A^2q_1, ..., A^iq_1] && (Aq_1 = \alpha q_1 + \beta q_2, \beta \neq 0) \\
&= [q_1, \alpha q_1 + \beta q_2, A(\alpha q_1 + \beta q_2), ..., A^{i-1}(\alpha q_1 + \beta q_2)] \\
&= [q_1, q_2, Aq_2, ..., A^{i-1}q_2] \\
&\vdots \\
&= [q_1, q_2, ..., q_{i-1}, Aq_i].
\end{aligned}$$

Consequently, Aq_i can be orthogonalized against $q_1, q_2, ..., q_i$ instead of A^iq_1 to obtain q_{i+1}. The observation leads to the Alg.(9) that was proposed by Arnoldi in 1951[19] to orthogonalize the Krylov subspace, for a non-Hermitian matrix A.

Algorithm 9 Arnoldi Iteration for Orthogonalizing the Krylov Subspace

Inputs: A, *initial vector* u, *subspace dimension* m

Output: Q_m, H_m

$q_1 = \frac{u}{\|u\|}$, $H_m = \mathbf{0}$ ▷ Initialize
for `j=1,2,...,m` **do**
  $w = Aq_j$
  for `i=1,2,...,j` **do** ▷ Gram-Schmidt orthogonalization
    $H_{i,j} = q_i^T w$
    $w = w - H_{i,j}q_i$
  end for
  $H_{j+1,j} = \|w\|$
  if $H_{j+1,j} = 0$ **then** ▷ An invariant subspace is found
    break
  end if
  $q_{j+1} = \frac{w}{H_{j+1,j}}$
end for
Return Q_m, H_m

One might notice that the Arnoldi process terminates if computed w vanishes because it means an invariant subspace is found.

The algebraic representation of the process can also be depicted. Let

$$Q_k = \begin{pmatrix} q_1 & q_2 & ... & q_k \end{pmatrix}$$

and

$$H_k = \begin{pmatrix} h_{11} & h_{12} & ... & h_{1,k-1} & h_{1k} \\ h_{21} & h_{22} & ... & h_{2,k-1} & h_{2k} \\ & h_{32} & \ddots & h_{3,k-1} & h_{3k} \\ & & \ddots & \vdots & \vdots \\ & & & h_{k,k-1} & h_{kk} \end{pmatrix}.$$

Then the Arnoldi process can be expressed by

$$AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^T$$

where

$$Q_k^T Q_k = I \quad \text{and} \quad Q_k^T q_{k+1} = 0.$$

Accordingly, it can be inferred that the H matrix is in upper Hessenberg form. Thus, this implies that in situations where A is a Hermitian matrix, the H matrix is also Hermitian, leading to a tridiagonal matrix. Therefore, the orthogonalization process of Krlyov subspace can be further improved by utilizing the structure of H, leading to the Alg.(10) that was proposed by Lanczos [20].

Hence, the Lanczos process can be expressed by

$$AQ_k = Q_k T_k + \beta_k q_{k+1} e_k^T$$

where Q_k is as described above,

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \alpha_{k-1} & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{pmatrix},$$

$$Q_k^T Q_k = I \quad \text{and} \quad Q_k^T q_{k+1} = 0.$$

Algorithm 10 Lanczos Iteration for Symmetric Eigenvalue Computation

Inputs: A, *initial vector* u, *subspace dimension* m

Output: V_m, T_m

$v_1 = \frac{u}{\|u\|}$, $\beta_0 = 0$ ▷ Initialize

for `j=1,2,...,m` **do**

  $w = Av_j$

  $\alpha_j = v_j^T w$

  $w = w - \alpha_j v_j - \beta_{j-1} v_{j-1}$

  $\beta_j = \|w\|$

  if $\beta_j = 0$ **then** ▷ An invariant subspace is found

    break

  end if

  $v_{j+1} = \frac{w}{\beta_j}$

end for

Form tridiagonal matrix T_m with α_i as diagonal elements and β_i as subdiagonal and superdiagonal elements

Return $V_m = [v_1, v_2, ..., v_m]$, T_m

As mentioned previously, both methodologies, Arnoldi and Lanczos, discussed above are techniques for orthogonalizing a Krylov subspace. Nevertheless, these methods can be used in the eigenpair computation by utilizing the fact that similar matrices have the same eigenpairs. Notice that A and H/T are similar matrices due to $QQ^T = I$ if the methods are iterated a number of times equal to the dimension of A. Hence, iterating the Krylov subspace method enough times to derive a matrix with reduced dimensions and subsequently computing its eigenpairs constitutes the approach for computing eigenpairs from the Krylov subspace. This approach's convergence behavior has been widely investigated, refer to [21, 22, 23, 24, 25] for deeper analysis.

Even though there have been various subspace-based approaches for the eigenpair computation, it is not the only paradigm for the task. In the following subsection, methods having different ways of approaching the problem are demonstrated.

2.3 Minimization Based Methods

Although the section is titled "Minimization-Based Methods," the specific objectives of the minimization problems are different. The title has been chosen to maintain consistency within the categorization.

2.3.1 Trace Minimization Eigensolver

TraceMin method was proposed by Sameh and Wisniewski [26] for solving the generalized symmetric eigenvalue problem and further improved in [27]. The method's main kernel is dependent on the following theorem.

Theorem 1 (Sameh and Wisniewski [26]) *Let A and B be $n \times n$ real symmetric matrices, with positive definite B, and let $\mathcal{X}$ be the set of all $n \times m$ matrices X for which $X^T BX = I_m$, $1 \leq m \leq n$. Then*

$$\min_{X \in \mathcal{X}} tr(X^T AX) = \sum_{i=1}^{m} \lambda_i,$$

where $\lambda_1 \leq \lambda_2 \leq ... \lambda_m < \lambda_{m+1} \leq ... \leq \lambda_n$ are the eigenvalues of $Ax = \lambda Bx$.

The important observation is the set of eigenvectors corresponding to the eigenvalues of the smallest magnitude forms the block of vectors X that resolves the constrained minimization problem. Accordingly, TraceMin algorithm can be derived by the following iterative scheme: assuming X_k is the approximation for the eigenvectors at a step, the algorithm updates it to X_{k+1} in a way that $tr(X_{k+1}^T A X_{k+1}) < tr(X_k^T A X_k)$. Hence, X_{k+1} is a better approximation for the eigenvectors.

X_k is corrected by using the $n \times m$ matrix Δ_k which is obtained by solving the following constrained optimization problem

$$\begin{aligned} &\text{minimize} && \text{tr}\left((X_k - \Delta_k)^T A (X_k - \Delta_k)\right) \\ &\text{subject to} && X_k^T B \Delta_k = 0. \end{aligned}$$

In the case of having symmetric positive definite A, this is equivalent to solving m independent problems of

$$\begin{aligned} &\text{minimize} && (x_{k,i} - d_{k,i})^T A (x_{k,i} - d_{k,i}) \\ &\text{subject to} && x_{k,i}^T B \Delta_k = 0 \end{aligned}$$

where subscript i corresponds to i^{th} column.

The mentioned constrained minimization problem can be transformed into an unconstrained minimization problem by using Lagrange's theorem, leading to the following saddle point problem for Δ_k

$$\begin{bmatrix} A & BX_k \\ X_k^T B & 0 \end{bmatrix} \begin{bmatrix} \Delta_k \\ L_k \end{bmatrix} = \begin{bmatrix} AX_k \\ 0 \end{bmatrix}$$

where L_k represents the Lagrange multipliers. Alternatively, the problem can be rewritten as

$$\begin{bmatrix} A & BX_k \\ X_k^T B & 0 \end{bmatrix} \begin{bmatrix} V_{k+1} \\ \bar{L}_k \end{bmatrix} = \begin{bmatrix} 0 \\ I_m \end{bmatrix} \tag{2.1}$$

by letting $V_{k+1} = X_k - \Delta_k$ and $\bar{L}_k = -L_k$. The described constrained optimization problem leads us to Alg.(11).

Algorithm 11 Trace Minimization Algorithm

Inputs: A, B, $U \in \mathbb{R}^{n \times m}$, *tolerance* ϵ, *maximum iterations max_iter*

Output: $\Lambda_{m,m+1,\ldots,n}$, $V_{m,m+1,\ldots,n}$

1: **for** `k=1,2,...,max_iter` **do**
2: $U = \mathrm{QR}_B(U)$ ▷ B-Orthonormalize U
3: $\hat{A} = U^T A U$ ▷ Form $\hat{A}$
4: $\hat{A}Y = \Lambda Y$ ▷ Approximate all eigenpairs of $\hat{A}$
5: $V = UY$ ▷ Compute Ritz vectors
6: **if** $\|AV - B\Lambda V\| < \epsilon$ **then**
7: **break**
8: **end if**
9: V ▷ Solve 2.1 approximately to obtain V
10: **end for**
11: **Return** Λ, V

For the convergence aspects and comparison with other well-known eigensolvers of TraceMin, refer to [28]. Moreover, the possible parallelization of the method was analyzed [29, 30].

2.3.2 Locally Optimal Block Preconditioned Conjugate Gradient Method

The locally optimal preconditioned conjugate gradient method performs an iterative maximization (minimization) of the generalized Rayleigh quotient

$$r(x) = r(A, B; x) = \frac{x^T A x}{x^T B x}$$

to find the largest (smallest) eigenpairs of $Ax = \lambda Bx$ [31]. The eigenvector residual

$$p = Ax - r(x)Bx$$

is proportional to the gradient of the generalized Rayleigh quotient, the direction of the steepest ascent. Accordingly, an iterative method can be defined as

$$x_{i+1} = x_i + \alpha_i w_i$$

where T is the preconditioner, $w_i = T(Ax_i - r(x_i)Bx_i)$ and α_i is the step size. The optimal step size can be obtained by maximizing (minimizing) the Rayleigh quotient.

Moreover, adding one extra vector to accelerate the convergence of the method leads to the following three-term recurrence of the method

$$x_{i+1} = \underset{y \in \text{span}\{x_i, w_i, x_{i-1}\}}{\text{argmin}} \; r(x) = \frac{x^T A x}{x^T B x}.$$

The solution to the problem is computed using the Rayleigh-Ritz procedure, leading to the computation of τ and γ coefficients. The method is given in Alg.(12).

Algorithm 12 Locally Optimal Preconditioned Conjugate Gradient Method

Inputs: A, B, *initial vector* u, *tolerance* ϵ, *maximum iterations max_iter*

Output: λ_1, v_1

$u_1 = u$
$p_1 = 0$
for `k=1,2,...,max_iter` **do**
    $\rho_k = (u_k^T B u_k)/(u_k^T A u_k)$
    $r_k = B u_k - \rho_k A u_k$
    $w_k = T r_k$
    Run Rayleigh Ritz for $(B - \rho_k A)$ *on Span*$\{w_k, u_k, p_k\}$
    $u_{k+1} = w_k + \tau_k u_k + \gamma_k p_k$
    if $\|A u_k - \rho_k B u_k\| < \epsilon$ **then**
        break
    end if
    $p_{k+1} = w_k + \gamma_k p_k$
end for
Return ρ, u

One should notice that this is only for one eigenpair computation, to obtain more than one eigenpair, a block version of the method is required to be used. For further implementation details, refer to [32].

The demonstration of the method concludes the literature review of approaches except FEAST. Followingly, in the next subsection FEAST eigensolver is detailed.

2.4 FEAST

Originally, FEAST eigensolver was proposed by Eric Polizzi in 2009 as a numerical eigensolver to find the eigenpairs of a Hermitian eigenvalue problem lying inside an interval $\mathcal{I} = [\lambda_-, \lambda_+]$, taking its inspiration from density matrix representations in quantum mechanics and the contour integration techniques [9]. However, this proposal lacked the theoretical analysis of FEAST. Therefore, Tang and Polizzi published another study in which they demonstrated that the FEAST algorithm can be viewed as an accelerated subspace iteration algorithm utilizing the Rayleigh-Ritz procedure in 2014 regarding the theoretical aspects. It is worth noting that the term acceleration refers to filtering where a function is applied to the matrix of interest to speed up the process. The algorithm's novelty is that the acceleration process utilizes an approximated spectral projector for the eigenspace desired [33]. Following this explanation, the FEAST algorithm can also be placed under "Subspace Iteration Methods" 2.2.2. Nonetheless, as it is the main focus of the study, the algorithm is detailed in this separate section. To have consistency with Tang and Polizzi's work, the FEAST algorithm is explained for the generalized Hermitian eigenvalue problems $Ax = \lambda Bx$. However, one should remember that this thesis focuses on symmetric standard eigenvalue problems, meaning $B = I$ and A is a real symmetric matrix.

2.4.1 Description

Let A and B be two $n \times n$ Hermitian matrices where B is positive definite, meaning $B = C^H C$ for an invertible C. As B is positive definite, B^{-1} exists which enables finding the eigenpairs of $M = B^{-1}A$ to obtain the eigenpairs of the generalized eigenvalue problem given by (A, B) pencil.

As mentioned, the FEAST algorithm is for the generalized eigenvalue problem and utilizes both an accelerator and Rayleigh-Ritz process. Accordingly, the subspace iteration with projection Alg.(8) should be modified to Alg.(13) so that the skeleton of FEAST is constructed. It is worth noting that without an accelerator, that is $p(M) = M$, the algorithm is the subspace iteration with projection for the generalized eigenvalue problem.

Algorithm 13 Accelerated Subspace Iteration with Rayleigh Ritz

Inputs: $A, B, U \in \mathbb{R}^{n \times m}$, *tolerance* ϵ, *maximum iterations max_iter*

Output: $\Lambda_{1,2,\ldots,m}$, $V_{1,2,\ldots,m}$

for `k=1,2,...,max_iter` **do**
  $W = p(M)U$ ▷ Compute the matrix product
  $W = QR$ ▷ Perform QR decomposition
  $\hat{A} = Q^T A Q$ ▷ Project A onto subspace
  $\hat{B} = Q^T B Q$ ▷ Project B onto subspace
  $\hat{A}V = \hat{B}\Lambda V$ ▷ Compute the eigenvalue decomposition of the pencil $(\hat{A}, \hat{B})$
  $U = QV$ ▷ Update subspace
  if $\|\hat{A}U - \hat{B}\Lambda U\| < \epsilon$ **then** ▷ Test for convergence
    break
  end if
end for
Return Λ, U

An ideal accelerator is $p(M) = X_{\mathcal{I}} X_{\mathcal{I}}^C B$, the spectral projector to the invariant subspace spanned by the corresponding eigenvectors of interest $X_{\mathcal{I}}$. In the case of having the ideal accelerator, the algorithm would converge in one iteration if $Y_1 = p(M)Q_0$ has full rank and the number of eigenvalues inside $\mathcal{I}$ is equal to the dimension of the initial subspace. In that point, [34] could be used for efficiently estimating the number of eigenvalues in an interval.

As indicated, having an ideal accelerator enables the algorithm to converge in one iteration; however, the paradoxical point is that the ideal accelerator utilizes the eigenvectors of the requested interval. Accordingly, the idea of the usage of the ideal accelerator can be perceived as there is a method that requires the eigenvectors of the requested interval to compute the eigenpairs of the interval. Consequently, one might argue that if the eigenvectors of the requested interval are already known, why would they be used to construct an accelerator instead of directly obtaining eigenvalues? This assessment inspires the novelty of the FEAST method which is the idea of approximating $p(M)$ instead of directly constructing it. Subsequently, the approximated $p(M)$, call $\tilde{p}(M)$, is used as an accelerator in Alg.(13), leading to the FEAST algorithm.

Following the previous explanation, a way to approximate the spectral projector is supposed to be constructed, meaning that $\tilde{p}(M)$ should be properly defined. As a beginning for the construction of the approximation function, start with a rational function that can be used in a contour integral manner and let $\tilde{p}(x) = \alpha/(\beta - x)$. Commonly, the function can be defined for $M = B^{-1}A$ as $\tilde{p}(M) = \alpha(\beta I - M)^{-1}$ [35]. As M is diagonalizable it can be decomposed to the $M = X\Lambda X^{-1}$ form and followingly

$$\begin{aligned} \tilde{p}(M) &= \alpha(\beta I - M)^{-1} \\ &= \alpha(\beta X X^{-1} - X\Lambda X^{-1})^{-1} \\ &= \alpha X(\beta I - \Lambda)^{-1} X^{-1} \\ &= X\tilde{p}(\Lambda)X^{-1} \end{aligned}$$

where $\tilde{p}(\Lambda)$ replaces each diagonal entry λ of Λ with $\tilde{p}(\lambda)$ as Λ is a diagonal matrix. Therefore, it can be concluded that for each eigenpair (λ, x) of M, $(\tilde{p}(\lambda), x)$ is an eigenpair of $\tilde{p}(M)$. Moreover, from the properties of the generalized eigenvalue problem, it is well-known that $X^{-1} = X^H B$. Accordingly,

$$\begin{aligned} \tilde{p}(M) &= X\tilde{p}(\Lambda)X^{-1} \\ &= X\tilde{p}(\Lambda)X^H B \end{aligned}$$

is the exact projector $X_{\mathcal{I}} X_{\mathcal{I}}^H B$ if $\tilde{p}$ is the indicator function of the interval $\mathcal{I}$.

The indicator function of the interval is given as

$$f(z) = \begin{cases} 1, & \text{if } z \in C \\ 0, & \text{otherwise} \end{cases}$$

where C is the circle centered at $c = (\lambda_+ + \lambda_-)/2$ with radius $r = (\lambda_+ - \lambda_-)/2$. Accordingly, it can be redefined by using Cauchy's integration formula

$$f(\lambda) = \frac{1}{2\pi i} \oint_C (z - \lambda)^{-1} dz.$$

Hence, $\tilde{p}$ can be interpreted as the indicator function of the interval, meaning that $\tilde{p}(M)$ is the exact spectral projector. However, since the eigenvalues are unknown, the integration is supposed to be approximated by numerical integration, leading to

an approximation of the spectral projector. Also, as line 2 of Alg.(13) includes multiplication with a U matrix, it can be added the approximation yielding

$$\tilde{p}(M)U = \sum_{k=1}^{q} \sigma_k(\phi_k B - A)^{-1}BU + \sum_{k=1}^{q} \bar{\sigma_k}(\bar{\phi_k} B - A)^{-1}BU \tag{2.2}$$

where $(w_k, t_k), k = 1, 2, ..., q$, are the q-point Gauss-Legendre rule of choice, $\phi(t) = c + re^{i\frac{\pi}{2}(1+t)}$, $\phi'(t) = i\frac{\pi}{2}re^{i\frac{\pi}{2}(1+t)}$, $\phi_k = \phi(t_k)$, $\sigma_k = w_k\phi'(t_k)/(2\pi i)$, and the 2q poles of $\tilde{p}$ are ϕ_k and $\bar{\phi_k}$ for $k = 1, 2, 3, ..., q$. Moreover, the computation can be further improved as follows if A, B, and U are real-valued

$$\tilde{p}(M)U = 2\sum_{k=1}^{q} \text{Re}\left(\sigma_k(\phi_k B - A)^{-1}BU\right).$$

Refer to [33] for the detailed derivation of the numerical integration.

Accordingly, the substitution of 2.2 to line 2 of Alg.(13) leads to the FEAST algorithm Alg.(14). It should be noted that lines 4 and 5 of the Alg.(13) are also performed in the FEAST algorithm; however, they are not included to provide a more concise scheme.

Algorithm 14 FEAST Eigensolver

Inputs: A, B, $U \in \mathbb{R}^{n\times m}$, *tolerance* ϵ, *maximum iterations max_iter*, C_p *Contour parameters*

Output: $\Lambda_{m,m+1,...,n}$, $V_{m,m+1,...,n}$

1: **for** `k=1,2,...,max_iter` **do**
2: $W = \tilde{p}(M)U$ ▷ Compute the contour integral based on C_p and Eq.(2.2)
3: $\hat{A}V = \hat{B}\Lambda V$ ▷ Compute the eigenvalue decomposition of the reduced $(\hat{A}, \hat{B})$
4: $U = QV$ ▷ Update subspace
5: **if** $\|\hat{A}U - \hat{B}\Lambda U\| < \epsilon$ **then** ▷ Test for convergence
6: **break**
7: **end if**
8: **end for**
9: **Return** Λ, U

Note that the eigenpairs of M lying inside the desired interval are now the (highly) dominant eigenpairs of the $\tilde{p}(M)$ because of the definition of the indicator function. Moreover, the i^{th} eigenvalue of $\tilde{p}(\lambda)$ converges at the rate of $|\tilde{p}(\lambda_i)|/|\tilde{p}(\lambda_{m+1})|$ where m is the dimension of the initial subspace and λ_{m+1} is the eigenvalue with

the $(m+1)^{th}$ largest magnitude [33]. In contrast to the subspace iteration algorithm having $|\lambda_i|/|\lambda_{m+1}|$ convergence rate which is heavily dependent on the eigenvalue spectrum, FEAST's convergence rate can be arbitrarily increased by either increasing the number of quadrature points used in the computation of the contour integral leading to a more accurate result or by increasing the dimension of the initial subspace [36].

After the description of the algorithm, further analysis, given in the following subsection, is required to express the purpose of the thesis.

2.4.2 Dissecting the Algorithm for Thesis Statements

This subsection briefly explains the parameters that must be set while employing the FEAST algorithm and ways to determine them. One of the most apparent parameters is the size and choice of the initial approximated subspace. The theoretical convergence bound based on the initial subspace is given in [33]; moreover, the result of a practical experiment using $(I - X_{1:10}X_{1:10}^T)$ as an initial subspace for the smallest 300 eigenvalues is illustrated in [37]. Accordingly, it could be concluded that having a better approximation for the requested eigenvalues as an initial subspace accelerates the process. However, the literature lacks suggestions for ways to construct better approximations to utilize. For instance, the mentioned practical experiment uses the first 10 eigenvalues which are apriorily not known. Another parameter to be specified is the dimension of the initial subspace p. Assuming the number of the eigenvalues lying inside the given interval as e, p choice having $|\tilde{p}(\lambda_{p+1})|/|\tilde{p}(\lambda_e)| \ll 1$ property is desirable as it leads faster convergence. The general advice for satisfying it is to set p as $1.5e$ [33]. However, generally, e is unknown complicating the decision of p. Nevertheless, there is a study for counting eigenvalues inside the FEAST algorithm [38].

The type of quadrature scheme and the number of quadrature points for solving the contour integral given in Eq.(2.2) plays a crucial role in the FEAST algorithm. Various stuides [39, 40, 41] have investigated the effects of these parameters on the FEAST algorithm. Based on the findings it is not possible to say that one of them is the best. Accordingly, this study uses Gauss-Legendre, the suggested one in the original FEAST paper, as a quadrature scheme. Also, it analyzes the effects of the

number of quadrature points.

After deciding on the quadrature scheme, a possible improvement regarding the contour integral Eq.(2.2) is orthogonalizing W after the computation as it might be rank-deficient [33]. The first suggested benefit of the application of rank-revealing QR [42] or SVD is they can resolve the rank deficiency and be used for the dimension reduction of the provided initial subspace as these methods could reveal the number of eigenvalues lying in the interval; the second one is they, also the QR factorization, can prevent the occurrence of the spurious eigenvalues: eigenvalues that do not originally belong to the input matrix; however, are returned as output [33, 39, 43]. Nonetheless, the spurious eigenvalues could be detected via the residual error of computed spurious eigenpairs as they yield relatively higher errors than expected.

A small toy example is provided in the following subsection better to demonstrate the flow of FEAST and the above parameters.

2.4.3 Toy Example

The flow of the FEAST algorithm is demonstrated by using the matrix

$$A = \begin{bmatrix} 1.97 & -0.55 & -0.05 & -1.06 \\ -0.55 & 2.34 & -0.67 & -0.58 \\ -0.05 & -0.67 & 0.30 & 0.35 \\ -1.06 & -0.58 & 0.35 & 1.02 \end{bmatrix}$$

whose eigenvalues are $0.037, 0.0918, 2.655$, and 2.8462, and B is an identity matrix. The lower and upper bounds are provided as 0 and 0.1 respectively, and the smallest 2 eigenvalues are targeted. The initial matrix is given as follows

$$U = \begin{bmatrix} 0.64 & 0.44 & 0.79 \\ 0.72 & 0.73 & 0.17 \\ 0.47 & 0.99 & 0.03 \\ 0.33 & 0.68 & 0.80 \end{bmatrix}$$

meaning that it has a dimension of 3, and the number of quadrature points (Ne) is equal to 8. In that case, after one step of the FEAST algorithm, the spectral projector

approximation yields the matrix

$$\tilde{p}(M) = \begin{bmatrix} 0.2821 & 0.1841 & 0.0333 & 0.4099 \\ 0.1841 & 0.177 & 0.2496 & 0.2239 \\ 0.0333 & 0.2496 & 0.9153 & -0.1259 \\ 0.4099 & 0.2239 & -0.1259 & 0.629 \end{bmatrix}$$

having its eigenvalues as $1.0034, 1, 0$, and 0 where the actual spectral projector

$$p(M) = \begin{bmatrix} 0.2816 & 0.1835 & 0.0324 & 0.4094 \\ 0.1835 & 0.1764 & 0.2486 & 0.2233 \\ 0.0324 & 0.2486 & 0.9136 & -0.1269 \\ 0.4094 & 0.2233 & -0.1269 & 0.6283 \end{bmatrix}$$

has eigenvalues $1, 1, 0$ and 0. Hence, it can be seen that approximation has a slight deficiency. Here, by remembering that $\tilde{p}(\lambda)$ is expected to be the indicator function of $\mathcal{I} = [\lambda_-, \lambda_+]$, the quality of approximation for the indicator function can be evaluated, which maps given eigenvalues $0.037, 0.0918, 2.6550$, and 2.8462 to $1, 1.0034, 0$ and 0 (also the eigenvalues of $\tilde{p}(M)$) respectively. Furthermore, to evaluate the similarity between the matrices, a simple approach is to sum the absolute differences of their corresponding elements, i.e., $\sum_{i=1}^{4}\sum_{j=1}^{4} abs(\tilde{p}(M)_{i,j} - p(M)_{i,j})$. Using this method, a total difference of 0.0127 is obtained which can be called a similarity score. Followingly, the reduced matrices $\hat{A}$ and $\hat{B}$ can be computed by $\hat{A} = Q^T A Q$ and $\hat{B} = Q^T B Q$ where $Q = \tilde{p}(M)U$, leading

$$\hat{A} = \begin{bmatrix} 0.0968 & 0.1373 & 0.0748 \\ 0.1373 & 0.1973 & 0.0997 \\ 0.0748 & 0.0997 & 0.0743 \end{bmatrix} \text{ and } \hat{B} = \begin{bmatrix} 1.0806 & 1.5006 & 0.9185 \\ 1.5006 & 2.1503 & 1.1058 \\ 0.9185 & 1.1058 & 1.2129 \end{bmatrix}.$$

After that, the solution of $\hat{A}V = \hat{B}\Lambda V$ yields

$$V = \begin{bmatrix} -0.3713 & -0.3792 & -0.5265 \\ -0.0716 & -0.4139 & -0.5428 \\ 0.6388 & -0.7110 & 0.2377 \\ -0.6700 & -0.4236 & 0.6097 \end{bmatrix} \text{ and } \Lambda = \begin{bmatrix} 0.037 & 0 & 0 \\ 0 & 0.0918 & 0 \\ 0 & 0 & 0.0437 \end{bmatrix}$$

which very importantly means that 0.0437 is returned as a spurious eigenvalue. Nevertheless, the crucial observation is that these eigenpairs have their residual error, i.e, $||Av - \lambda v||$, as 2.5924×10^{-11}, 4.8005×10^{-12} and 2.6258 respectively. Additionally, another very important observation is that if QR factorization is applied to Q before the computation, then instead of 0.0437, 2.6696 is computed as an eigenvalue which prevents the appearance of the spurious, as it does not lie inside the search interval.

After demonstrating the flow, the importance of the number of quadrature points can be seen in Table (2.1), which clearly shows that as the number of quadrature points increases, the approximation gets better.

Metric / **Ne**	Similarity Score	Eigenvalue 1	Eigenvalue 2	Eigenvalue 3	Eigenvalue 4
2	1.0289	1.012	0.7224	-0.0001	-0.0001
4	0.0781	0.9997	0.979	0	0
8	0.0127	1.0034	1	0	0
16	9.3588×10^{-5}	1	1	0	0
32	4.1764×10^{-10}	1	1	0	0

Table 2.1: Metric Comparison of Different Number of Quadrature Points

The properties of FEAST mentioned in the previous subsection combined with the demonstration in this one provide the general perspective that is required to dive further into questions addressed in the thesis. Accordingly, these as well as the proposed methods are described in the following chapter.

CHAPTER 3

PROPOSED WORK

This chapter plays a crucial role in this thesis, it starts by indicating the central questions that are aimed to be answered. Later, each subsequent section depicts the algorithms obtained by improving the FEAST, and finally, the per-iteration requirements of these methods are given.

3.1 Summary of the proposed method

The following research questions are addressed in this thesis:

1. How important the provided initial subspace is? Is having extra computational preprocesses at the beginning of the FEAST algorithm in order to use a better initial approximation worth it overall?

2. Does having an extra orthogonalization for the W matrix at each step of FEAST reduce the number of iterations required for the algorithm to converge? If so, does it shorten the overall computation time? Additionally, how does this orthogonalization affect the occurrence of spurious eigenvalues?

3. Does combining the FEAST algorithm with the inverse subspace iteration and iteratively running them one after another affect the convergence?

Even though some of the questions have already been addressed by different studies as stated in the previous chapters, the main goal of the thesis is to answer these by considering the eigenvalue spectrum. The objective is to discover when these

modifications are required based on the spectrum. In the next sections, the proposed methods are briefly described.

3.2 PFEAST

The suggested way for providing a better initial guess to FEAST in this work is the usage of the (shifted) inverse subspace iteration Alg.(15) as a preprocessor, leading to Preprocessed FEAST (PFEAST) algorithm Alg.(16).

Algorithm 15 Shifted Inverse Subspace Iteration (SISI)

Inputs: A, σ, $Y \in \mathbb{R}^{n \times m}$, *tolerance* ϵ, *maximum iterations max_iter*

Output: $\Lambda_{m,m+1,\ldots,n}$, $V_{m,m+1,\ldots,n}$

$A - \sigma I = LU$ ▷ LU decomposition for inverse computation

for `k=1,2,...,max_iter` **do**

  $W = U^{-1}L^{-1}Y$ ▷ Solve

  $W = QR$ ▷ Perform QR decomposition

  $Y = Q$ ▷ Update subspace

  $\Lambda = Y^T AY$ ▷ Approximate eigenvalues

  if $\|AY - Y\Lambda\| < \epsilon$ **then** ▷ Test for eigenvalues

    break

  end if

end for

Return Λ, Y

Theoretically, as explained in the previous sections, FEAST has roots in the inverse subspace algorithm and has advanced over it. Therefore, in theoretical convergence analysis, the proposed method does not have any superiority over FEAST; however, as discussed in the experiment part, practically speaking, the suggested method might offer a computationally better way to solve the eigenvalue problem since an iteration of SISI costs less than that of FEAST.

Algorithm 16 PFEAST Eigensolver

Inputs: A, $U \in \mathbb{R}^{n \times m}$, *tolerance* ϵ, σ, *maximum iterations max_iter*, *sisi iterations sisi_iter*, C_p *Contour parameters*

Output: $\Lambda_{m,m+1,\ldots,n}$, V

$U = SISI(A, \sigma, U, \epsilon, \textit{sisi\_iter})$ ▷ PFEAST step: U is improved by SISI

for `k=1,2,...,max_iter` **do**

  $W = \tilde{p}(M)U$ ▷ Compute the contour integral based on C_p and Eq.(2.2)

  $\hat{A}V = \Lambda V$ ▷ Compute the eigenvalue decomposition of the reduced ($\hat{A}$)

  $U = WV$ ▷ Update subspace

  if $\|\hat{A}U - \Lambda U\| < \epsilon$ **then** ▷ Test for convergence

    break

  end if

end for

Return Λ , U

3.3 QFEAST

As mentioned earlier, for various reasons, the W matrix obtained after approximately computing the contour integral might lose its orthogonality, decelerating the convergence of FEAST. Accordingly, several options to further orthogonalize it have been proposed. The one used in this study to answer the second question mentioned at the beginning of the third chapter is QR factorization which leads to the QR FEAST (QFEAST) algorithm depicted in Alg.(17).

Algorithm 17 QFEAST Eigensolver

Inputs: A, $U \in \mathbb{R}^{n \times m}$, *tolerance* ϵ, *maximum iterations max_iter*, C_p *Contour parameters*

Output: $\Lambda_{m,m+1,\ldots,n}$, $V_{m,m+1,\ldots,n}$

1: **for** `k=1,2,...,max_iter` **do**
2: $W = \tilde{p}(M)U$ ▷ Compute the contour integral based on C_p and Eq.(2.2)
3: $W = QR$ ▷ QFEAST step: W is orthogonalized to improve stability
4: $\hat{A}V = \Lambda V$ ▷ Compute the eigenvalue decomposition of the reduced ($\hat{A}$)
5: $U = QV$ ▷ Update subspace
6: **if** $\|\hat{A}U - \Lambda U\| < \epsilon$ **then** ▷ Test for convergence
7: **break**
8: **end if**
9: **end for**
10: **Return** Λ , U

3.4 HFEAST

As also mentioned in the PFEAST section, FEAST has roots in the inverse subspace algorithm; therefore, alternatingly employing them to compute eigenvalues is not supposed to yield a numerically better algorithm. However, as a step of SISI is far less computationally costly than FEAST, practically speaking, adding an extra step to the end of each FEAST iteration might make the algorithm converge in a few iterations earlier, providing a relatively small performance improvement. This assumption leads to the Hybrid FEAST (HFEAST) algorithm Alg.(18).

Moreover, the HFEAST algorithm is further expanded by adding QR factorization inside as it is done to convert FEAST to QFEAST.

Algorithm 18 HFEAST Eigensolver

Inputs: $A, Y \in \mathbb{R}^{n \times m}$, *tolerance* ϵ, *maximum iterations max_iter*, C_p *Contour parameters*

Output: $\Lambda_{m,m+1,\ldots,n}$, $V_{m,m+1,\ldots,n}$

$A - \sigma I = LU$ ▷ HFEAST step: LU decomposition for inverse computation
for `k=1,2,...,max_iter` **do**
  $W = \tilde{p}(M)Y$ ▷ Compute the contour integral based on C_p and Eq.(2.2)
  $\hat{A}V = \Lambda V$ ▷ Compute the eigenvalue decomposition of the reduced ($\hat{A}$)
  $Y = WV$ ▷ Update subspace
  if $\|\hat{A}Y - \Lambda Y\| < \epsilon$ **then** ▷ Test for convergence
    break
  end if
  $W = U^{-1}L^{-1}Y$ ▷ HFEAST step: solve
  $W = QR$ ▷ HFEAST step: perform QR decomposition
  $Y = Q$ ▷ Update subspace
end for
Return Λ, Y

3.5 Per Iteration Comparisons of Methods

The above methods require different types of linear algorithms such as QR factorization, and linear system solutions because to compute Eq.(2.2) for each quadrature point a linear system is solved instead of taking matrix inverse. Vanilla versions of both have $\mathcal{O}(n^3)$ complexity. Nevertheless, there are studies [44, 45] to speed up the QR factorization, and a comprehensive overview of Sparse Linear Solvers can be found in [46]. Accordingly, Tbl.(3.1) compares computationally costly steps that each method requires in an iteration where Ne is the number of quadrature points. It should be noted that even though FEAST and PFEAST have the same number of operations per iteration, as PFEAST is preprocessed with SISI, in general, by assuming that they require the same number of iterations to converge, PFEAST has an extra cost of QR Factorization and Linear System Solution by the number of iterations that SISI is run before it. Accordingly, the convergence behaviors of methods over various

matrices are investigated in the following chapter.

Operation / **Method**	QR Factorization	Linear System Solution
SISI	1	1
FEAST	0	Ne
PFEAST	0	Ne
QFEAST	1	Ne
HFEAST	1	Ne+1
HFEAST with QR	2	Ne+1

Table 3.1: Per Iteration Cost Comparisons of Methods

CHAPTER 4

NUMERICAL EXPERIMENTS

The numerical experiments are conducted in two different setups. Firstly, methods are tested by using Synthetic smaller-sized data. Followingly, real-life datasets, which are sparse, from the SuitSparse Matrix Collection [47] are used for testing. The experiments are performed on an AMD Opteron 6376 system with 64 cores and 64GB L2 cache PC by using Julia [48] version "1.9.1".

4.1 Synthetic Data

As explained previously, one of the objectives of this thesis is to investigate the effect of proposed methods on a variety of eigenvalue spectrums. Hence, firstly 3 different sets of real eigenvalues corresponding to dense, discrete, and sparse spectrums, which will be explained shortly, are randomly generated to compare the performances of 5 different methods over finding the smallest eigenvalues. Followingly, a random real symmetric matrix that has the provided list of real eigenvalues as its eigenvalues are constructed by Alg.(19).

Algorithm 19 Random Real Symmetric Matrix Generator

Inputs: *List of real numbers having a length of n* L

Output: *A real symmetric matrix having these real numbers as eigenvalues* A

1: $A = rand(n, n)$ ▷ Generate a random matrix from numbers between -1 and 1
2: $Q = QR(A)$ ▷ Compute QR factorization
3: $\hat{A} = Qdiag(L)Q^T$ ▷ Generate a matrix having the eigenvalues
4: $A = \frac{1}{2}(\hat{A} + \hat{A}^T)$ ▷ Ensure symmetricity
5: **Return** A

Dense Spectrum: The dense spectrum corresponds to the eigenvalue distribution in which the eigenvalues have two clusters; for the search interval and the others, but those two clusters are close to each other. Fig.(4.1) can depict the dense spectrum's abstract look.

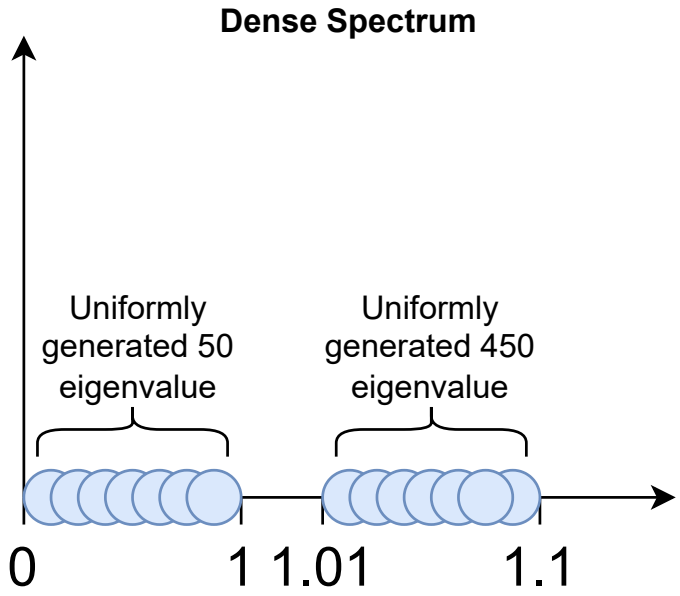


Figure 4.1: The abstract overview of the dense spectrum

Discrete Spectrum: Unlike to the dense one, the discrete spectrum represents cases where the closest eigenvalue after the search interval is placed relatively far away. Fig.(4.2) is an illustration of this case.

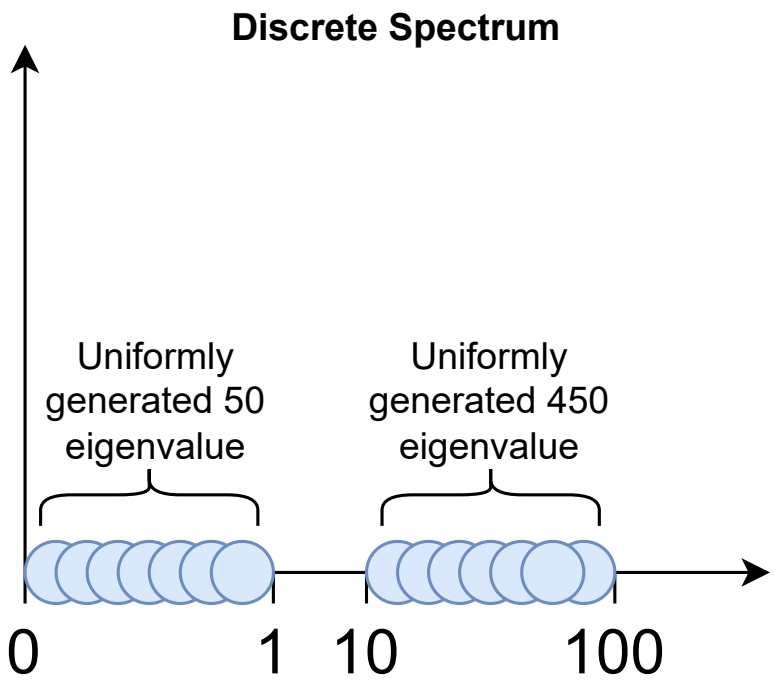


Figure 4.2: The abstract overview of the discrete spectrum

Sparse Spectrum: Finally, the sparse spectrum is for when the closest eigenvalue is relatively near to the boundary; however, the outside distribution is sparse which can be demonstrated by Fig.(4.3).

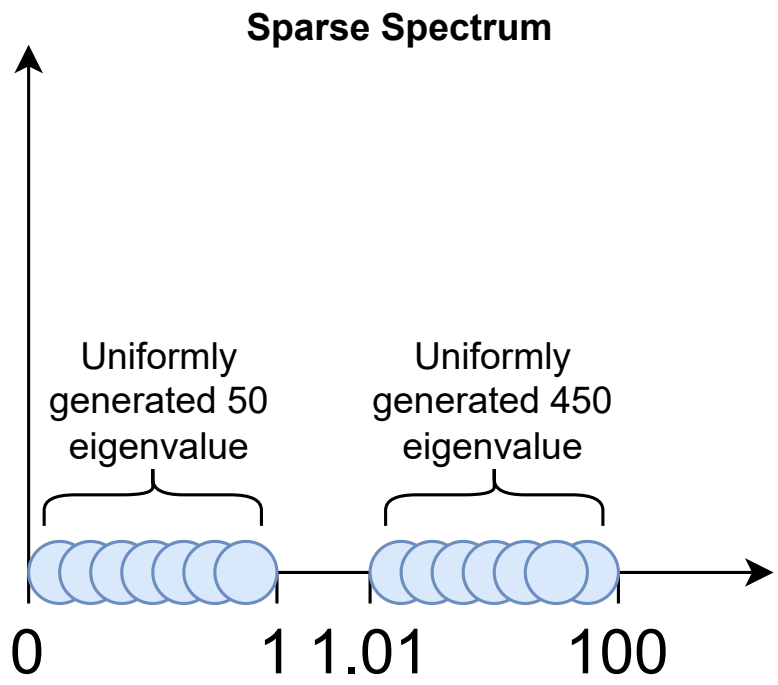


Figure 4.3: The abstract overview of the sparse spectrum

Experimental Setup: Each algorithm is executed 10 times with different initial matrices to decrease the effects of randomness across all setups, and the results of these are provided by using the box-plot explained in Fig.(4.4). In the figure, the Q1 lower quartile is the 25th percentile of the data, meaning 25% of the data points are below this value, and the Q3 is the 75th percentile, meaning 75% of the data points are below this value. Accordingly, the interquartile range (IQR) is calculated by Q3-Q1. Following that, the points greater than the upper fence and the ones less than the lower fence are classified as outliers.

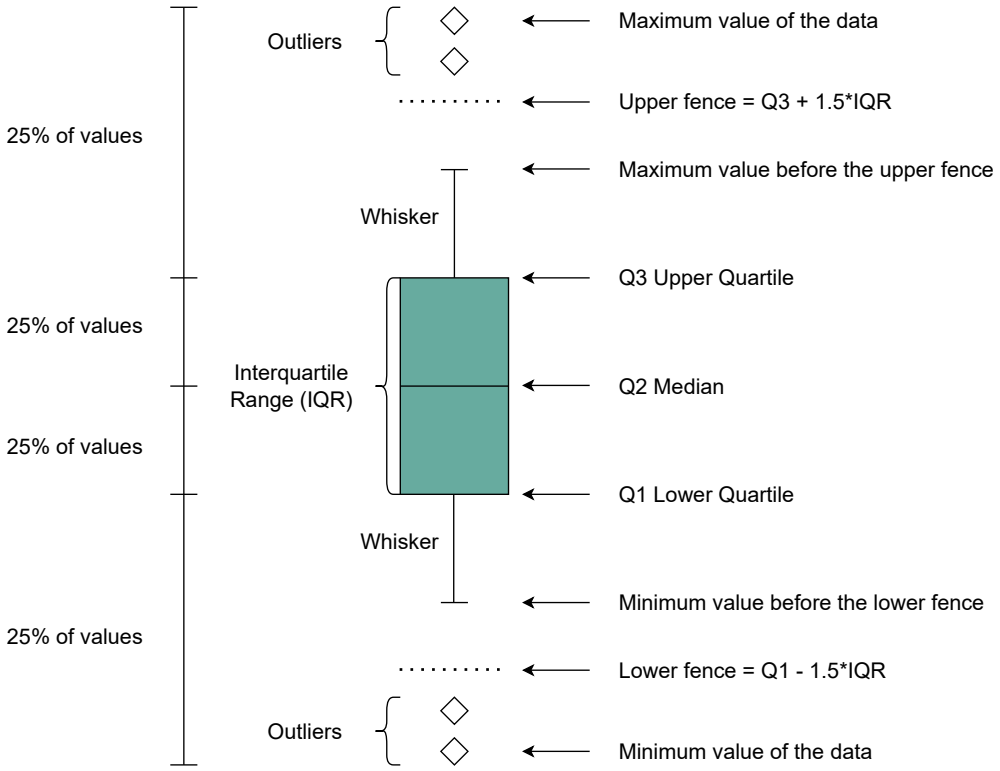
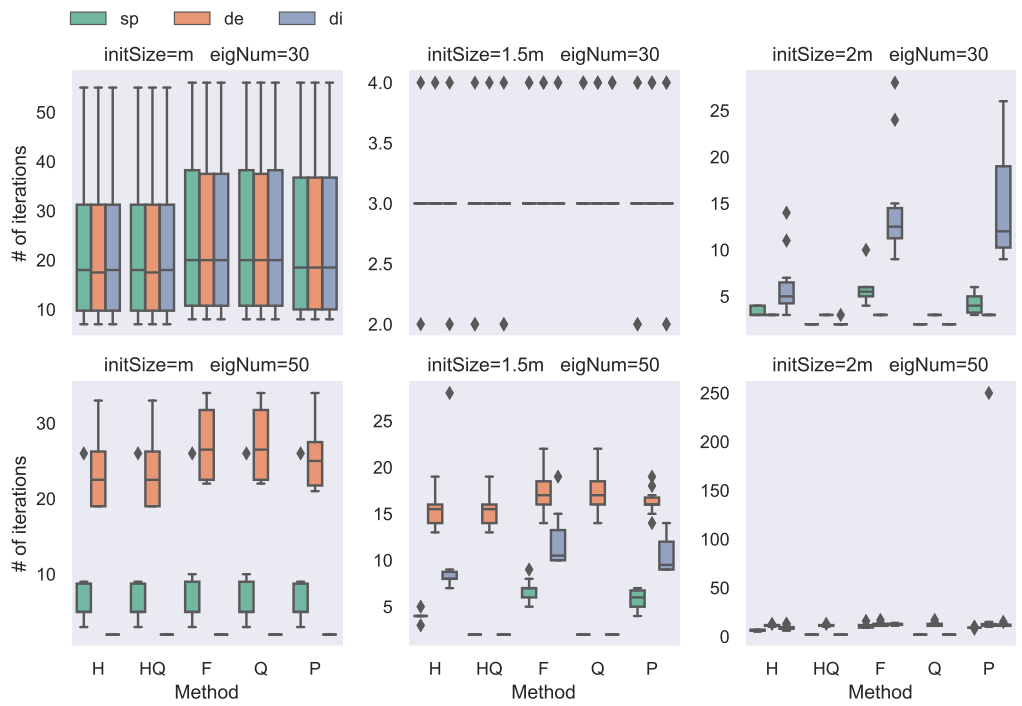


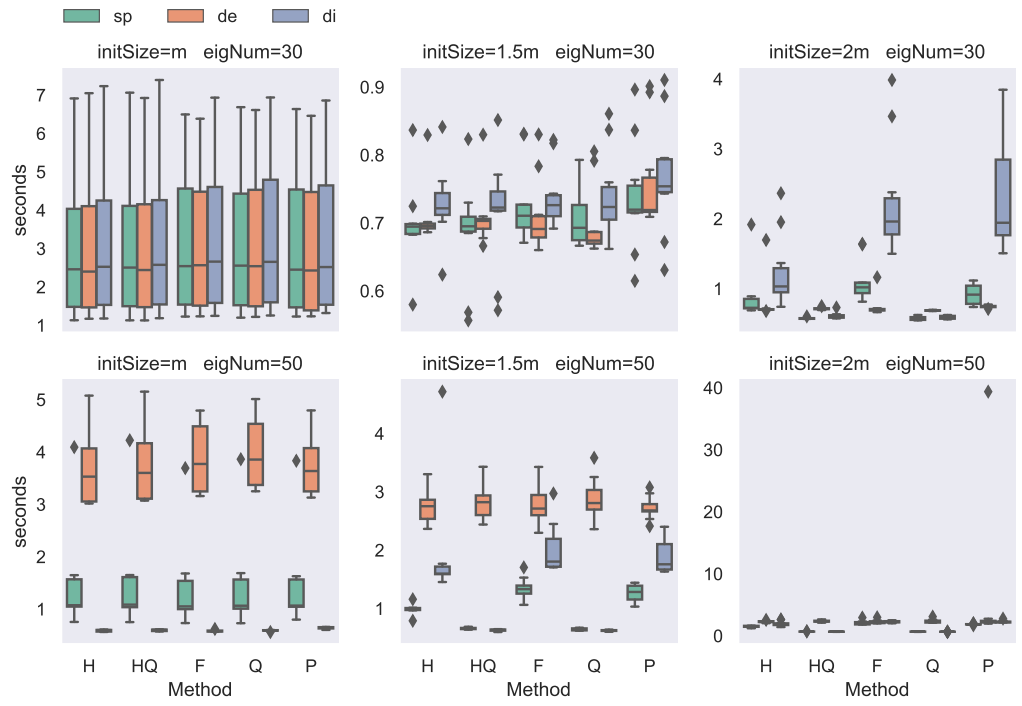
Figure 4.4: The explanation of box-plot

In the test setup, two desired numbers of eigenvalues, 30 and 50, are used. For 50 the search interval is provided as $[0, 1]$. For 30 the lower bound is given as 0 too but the upper bound is provided as the arithmetic mean of λ_{30} and λ_{31}. The dimension of the initial subspace is determined by expanding the desired number of eigenvalues by factors of 1, 1.5, and 2, so for instance, if the desired number of eigenvalues is 30 and the factor parameter is $1.5m$, then the dimension of the initial subspace is 45. Also, 2,4,8,16, and 32 are used as the number of quadrature points, Ne, to determine its effect over convergence. The maximum number of iterations allowed is 250. In the figures, F corresponds to the original FEAST algorithm Alg.(14), Q corresponds to the QFEAST algorithm Alg.(17), H corresponds to HFEAST algorithm Alg.(18), HQ corresponds to the Hybrid FEAST with QR algorithm in which QFEAST algorithm is further enhanced by applying the Hybrid approach and finally P corresponds to the PFEAST Alg.(16). Moreover, it should be noted that the stopping criteria for algorithms is the maximum of the residual errors of the smallest m eigenvalues

where m is the number of eigenvalues to compute. Namely, $||Av_i - v_i\lambda_i||_2$ is computed for each eigenvalue and m^{th} smallest of them is taken. If that one is less than 10^{-13}, the algorithm stops. This approach is done to eliminate the effect of spurious eigenvalues on the convergence of algorithms because in certain cases the suggested stopping criteria with the appearance of the spurious eigenvalues caused algorithms not to terminate. Nevertheless, one of the widely used techniques to detect spurious eigenvalues is checking their residual error since errors tend to be relatively higher than normal as they are not real members of the matrix. The essential point, which is important for all the following figures of this chapter, is that for the number of iterations of PFEAST, the (S)ISI precedence phase is not included in the plots; however, the total duration plots of the PFEAST contains the initial phase.



a) Number of iterations

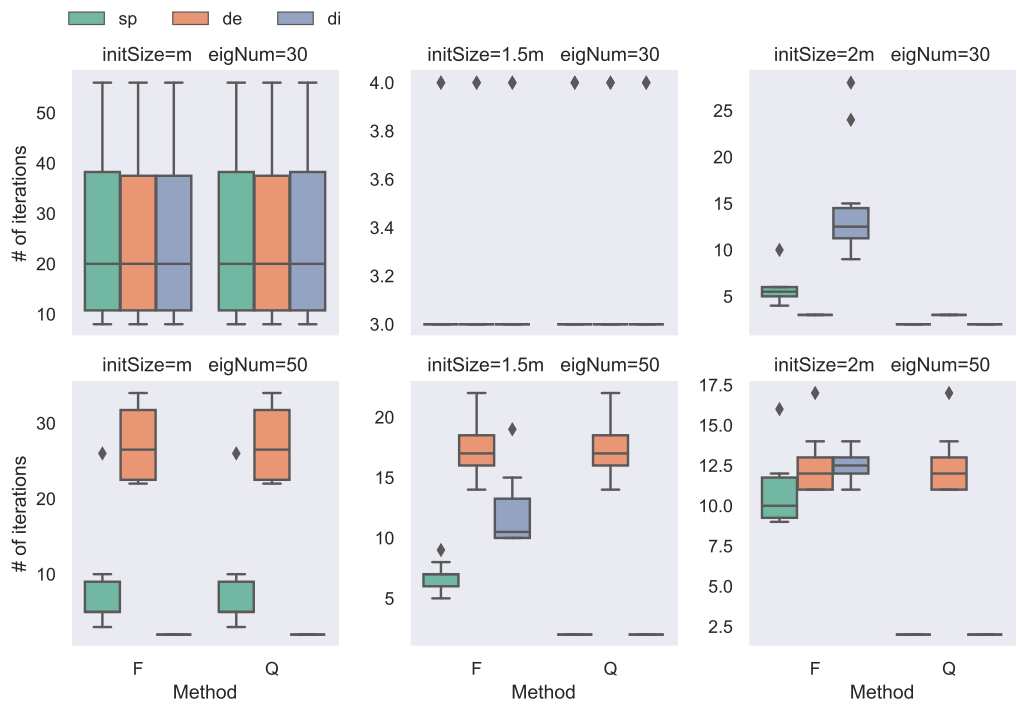


b) Time in seconds

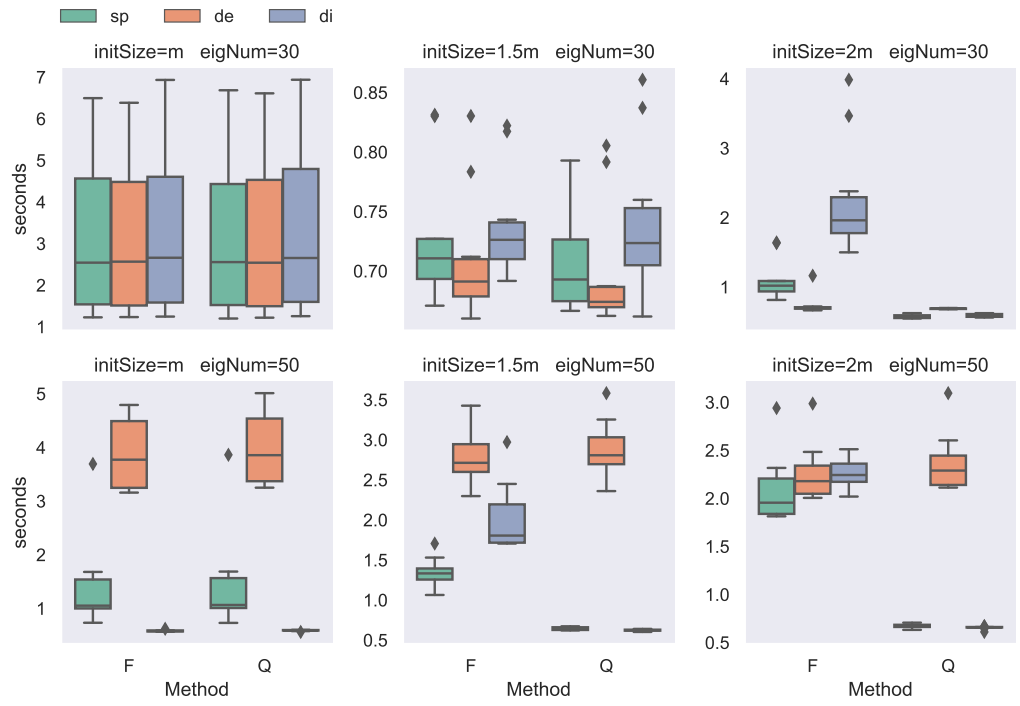
Figure 4.5: Overall test results for the synthetic data where the smallest eigenvalues are computed and Ne is 8

Fig.(4.5) shows the results of all tests where Ne is fixed to 8 in a comprehensive summary. Figures a part gives the number of iterations algorithms required, and b part shows the duration of them. In the figure, the x-axis for each sub-plot corresponds to the different methods while the y-axis indicates the number of iterations / the total duration each algorithm requires to terminate. Further, the overall figure's y-axis (called eigNum) corresponds to the number of eigenvalues desired to compute whereas the x-axis corresponds (called initSize) to the dimension of the initial subspace provided, meaning that they are used to categorize the sub-plots. Finally, colors are used to show the spectrum types.

The figure shows, which is also valid for the following experiments, in general, the patterns of boxplots for the number of iterations and the total times are quite similar. Moreover, when eigNum is set to 30 and initSize is either m or $1.5m$, there are minimal differences in computation times across the methods. However, for other configurations, more meaningful comparisons between the methods can be drawn. To navigate through further observations in more detail, on the following pages same results are depicted on more detailed figures.



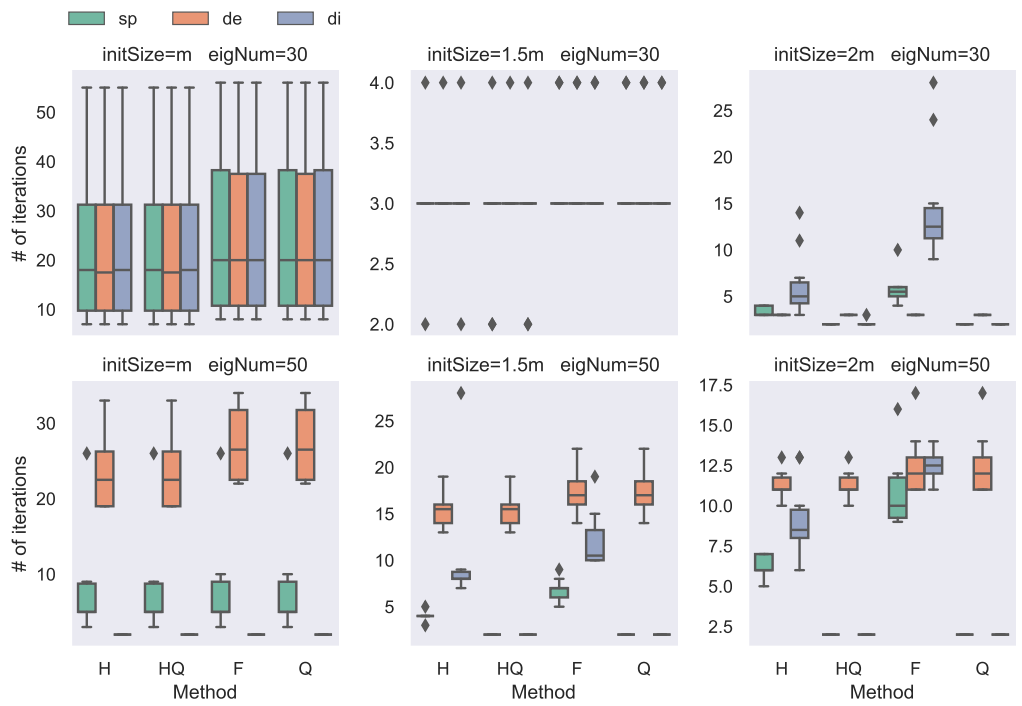
a) Number of iterations



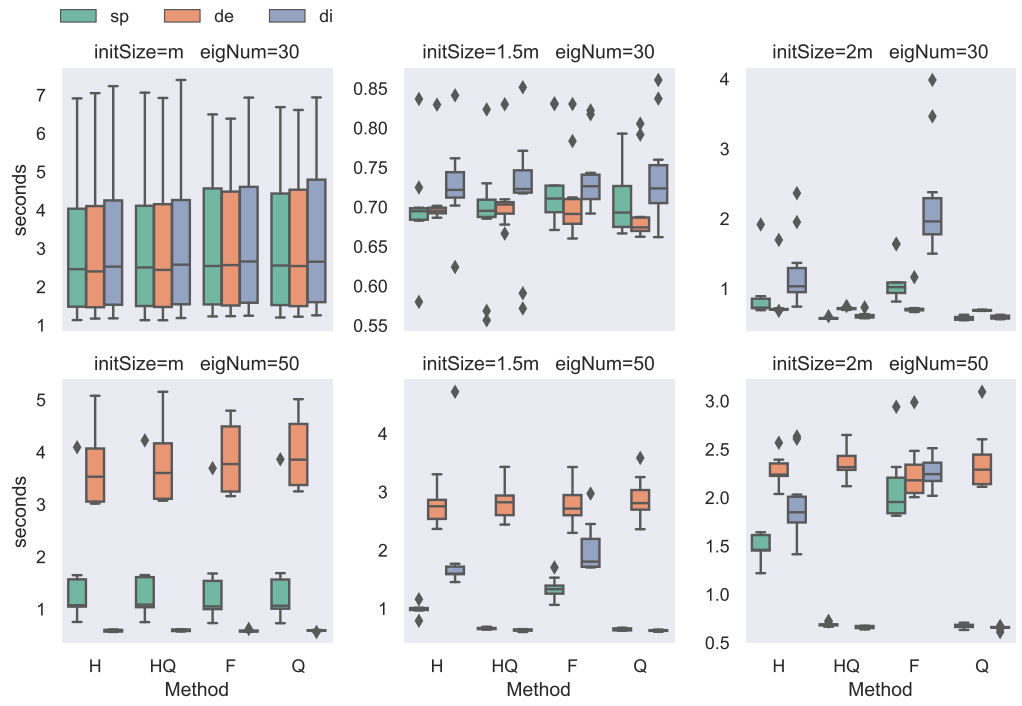
b) Time in seconds

Figure 4.6: Test results of FEAST and QFEAST for the synthetic data where the smallest eigenvalues are computed and Ne is 8

Fig.(4.6) compares the performances of FEAST and QFEAST. The first observation is that in certain cases QFEAST converges faster in terms of both the number of iterations and total time than FEAST, and in others, they take the same number of iterations; moreover, in the setups that they required the same number of iterations, not much time difference can be observed. Based on that, it could be safe to say that preferring QFEAST over FEAST occasionally is a better way to compute; furthermore, as mentioned, for the setups that QFEAST is not superior, the addition of the extra QR factorization does not significantly slow the convergence in terms of the total duration. Consequently, choosing it over FEAST might be a safer approach as sometimes the algorithm could be speed-upped, and if not, the loss is not weighty. After that conclusion, the cases where the speed up gained has to be identified. Note that there is no performance improvement for the dense spectrum and in other spectrums, improvement happens if the dimension of the initial matrix (initSize x eigNum) is larger than 50, the number of eigenvalues lying inside $[0, 1]$. As a result, if the distribution of the first k eigenvalues where k is the dimension of the initial subspace has a gap like in the discrete spectrum, applying the additional QR factorization leads to fewer iterations to convergence. Moreover, by examining the gains for both sparse and discrete spectrums, it could be said that as the length of the gap increases, performance improvement also advances.



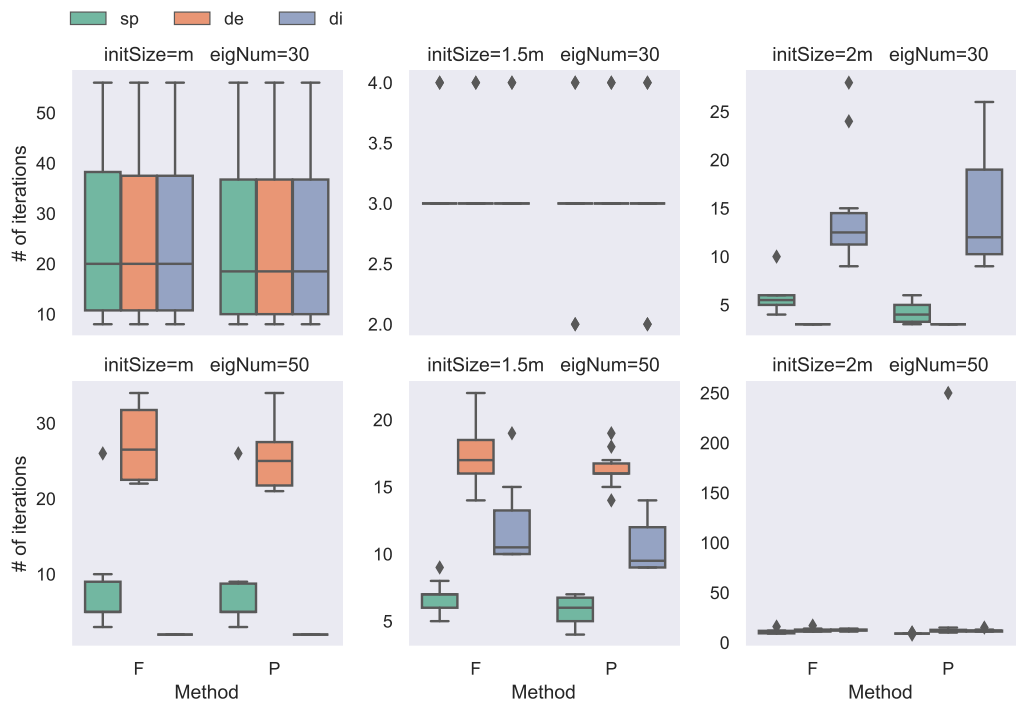
a) Number of iterations



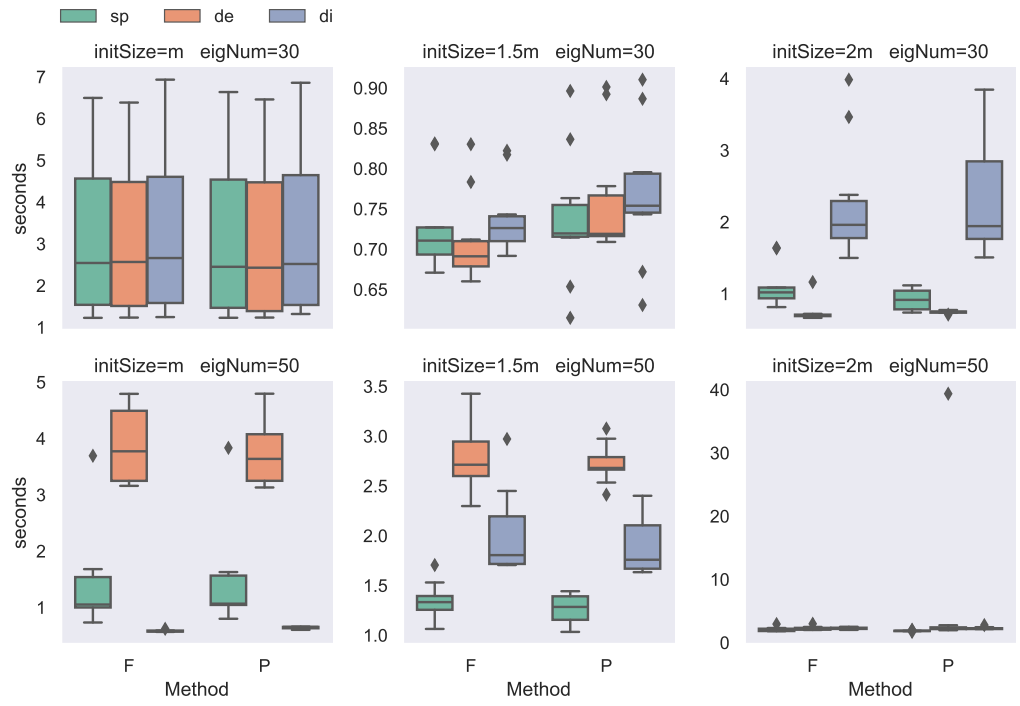
b) Time in seconds

Figure 4.7: Test results of FEAST, HFEAST, QFEAST, and Hybrid QFEAST for the synthetic data where the smallest eigenvalues are computed and Ne is 8

After investigating the differences between QFEAST and FEAST, the next comparison includes the Hybrid methods given in Fig.(4.7). An interesting observation might be that in contrast to QFEAST, Hybrid methods seem to work better in terms of the number of iterations for the dense spectrum in which Hybrid ones are never worse. Even though for other spectrums in certain cases Hybrid QFEAST is better than HFEAST, they have the same number of iterations for all dense spectrum settings. However, in terms of the total duration, they start to be slower than others as the dimension of the initial subspace increases. Consequently, by remembering that the extra QR factorization does not have an observable effect on the dense spectrum, it can be concluded that Hybrid approaches are the favorite ones in these distributions in terms of the number of iterations as they generally converge a few iterations earlier; however, this gain is rooted for the extra ISI step, which owns its additional computational cost, employed in them hence Hybrid ones are not the superior in terms of the total duration which is affected by the dimension of the initial subspace.



a) Number of iterations

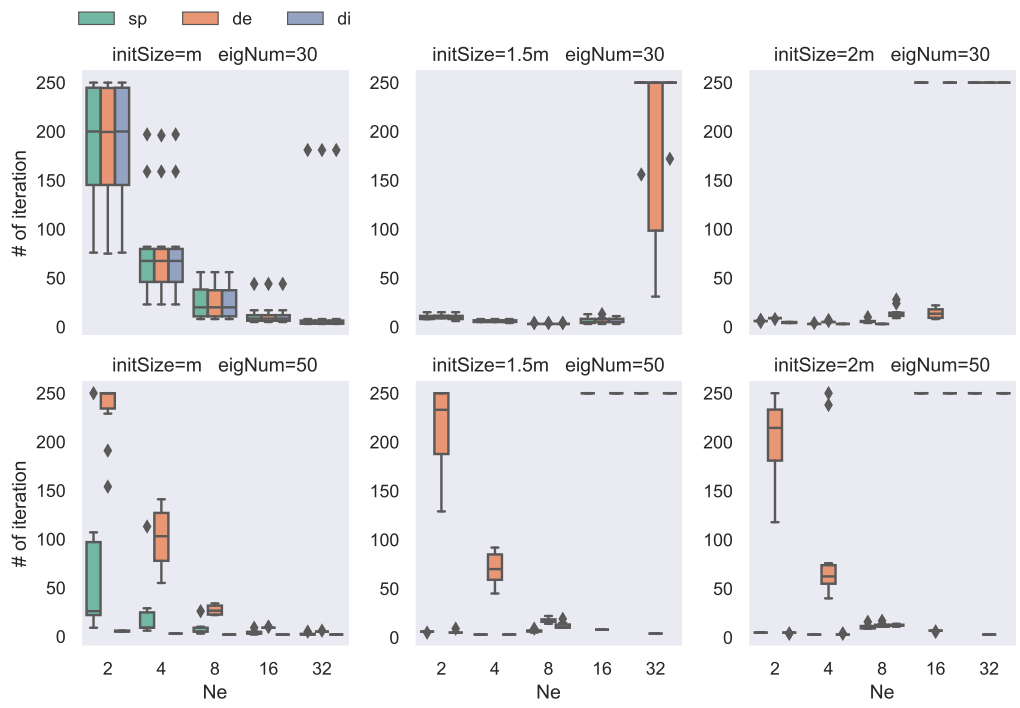


b) Time in seconds

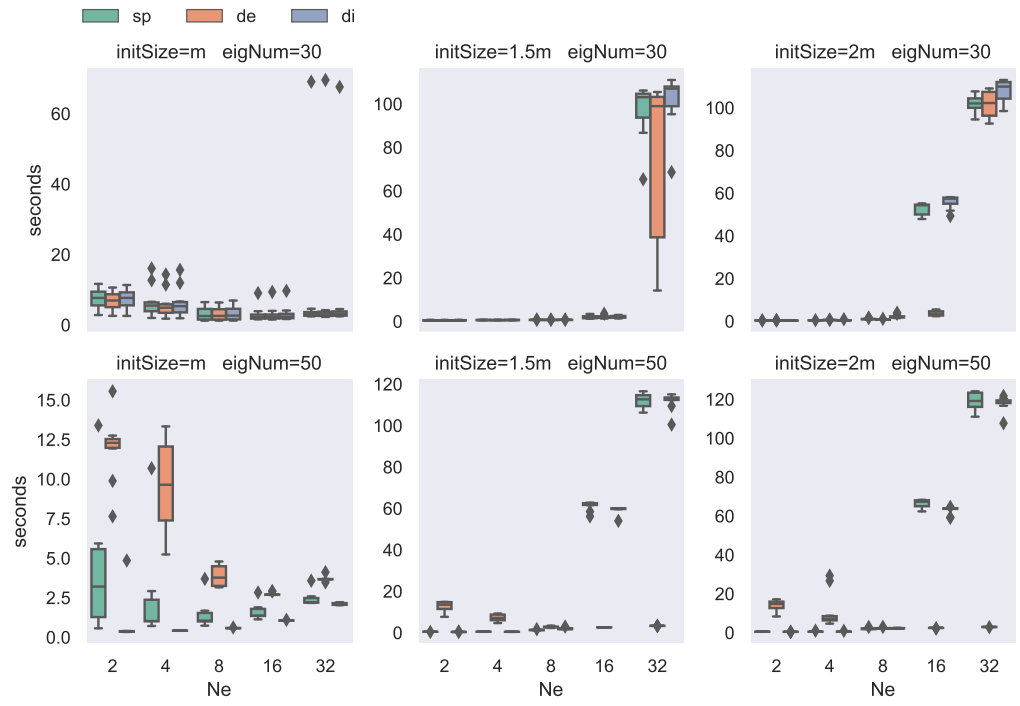
Figure 4.8: Test results of FEAST and PFEAST for the synthetic data where the smallest eigenvalues are computed and Ne is 8

The remaining comparison is the effect of performing ISI as a preprocessor, shown in Fig.(4.8). By keeping in mind that the ISI precedence phase is not reflected in the number of iterations figure, PFEAST can be said to converge a few iterations earlier than FEAST; however, in terms of the total duration, unsurprisingly, it is not possible to label one as a superior as they have the same theoretical convergence rate.

After the finalization of the per-method comparisons, the effect of Ne also has to be investigated, which is done by evaluating FEAST and QFEAST with different Ne values in Fig.(4.9) and Fig.(4.10) respectively. One should pay attention that the x-axis for each subplot corresponds to the number of quadrature points. Analyzing the FEAST figure reveals that initially, increasing the number of quadrature points accelerates the algorithm in terms of the number of iterations and the total computation time. However, when the number of quadrature points reaches 16 and 32, the algorithm fails to converge. Subsequently, the investigation of the QFEAST shows that the increase of Ne does not have a negative effect on the number of iterations; whereas, for higher quadrature points, it slows the convergence time. Possibly, the reason for that is higher quadrature numbers cause the loss of orthogonality due to the increasing number of floating point errors, which is resolved in QFEAST by the costly extra QR factorization. Finally, even though increasing the number of quadrature points increases the accuracy of the approximation and hence speeds up the convergence in terms of the number of iterations, it also increases the per-iteration cost of algorithms, making it not possible to detect the best number of quadrature points which is confirmed by the plots in which there is no favorite value for the number of quadrature points.

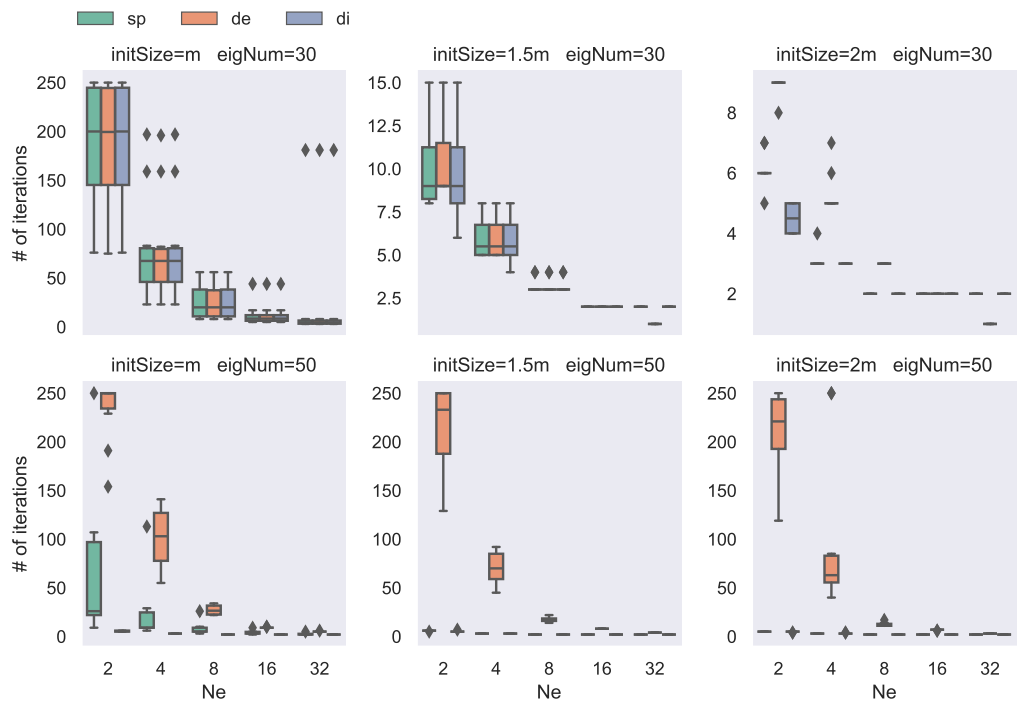


a) Number of iterations

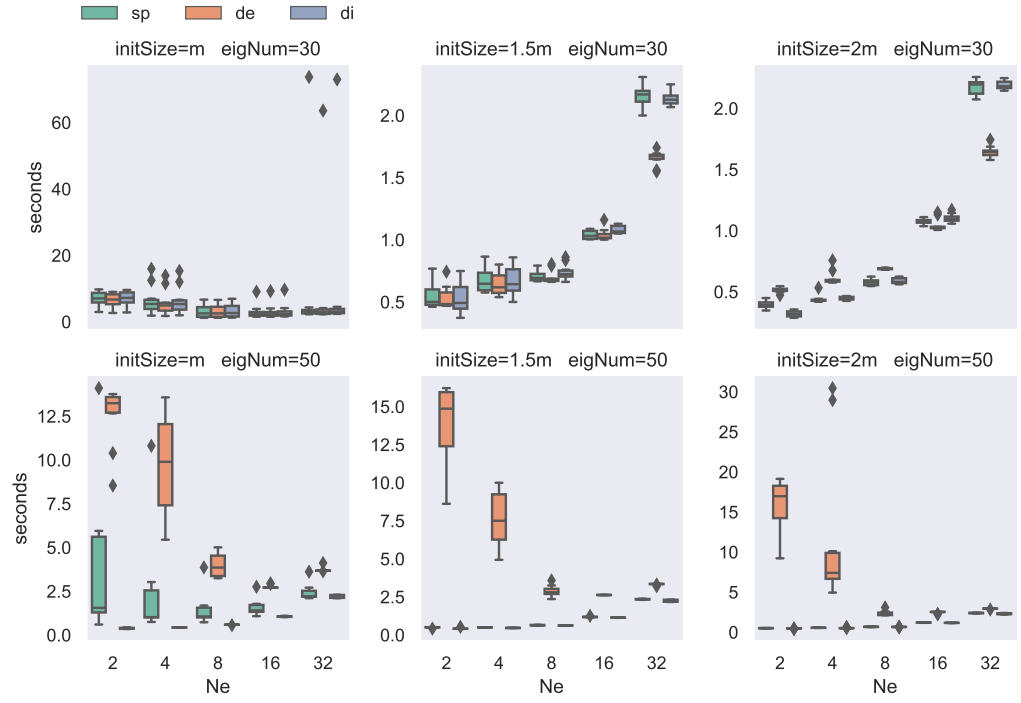


b) Time in seconds

Figure 4.9: Test results of FEAST for the synthetic data where the smallest eigenvalues are computed



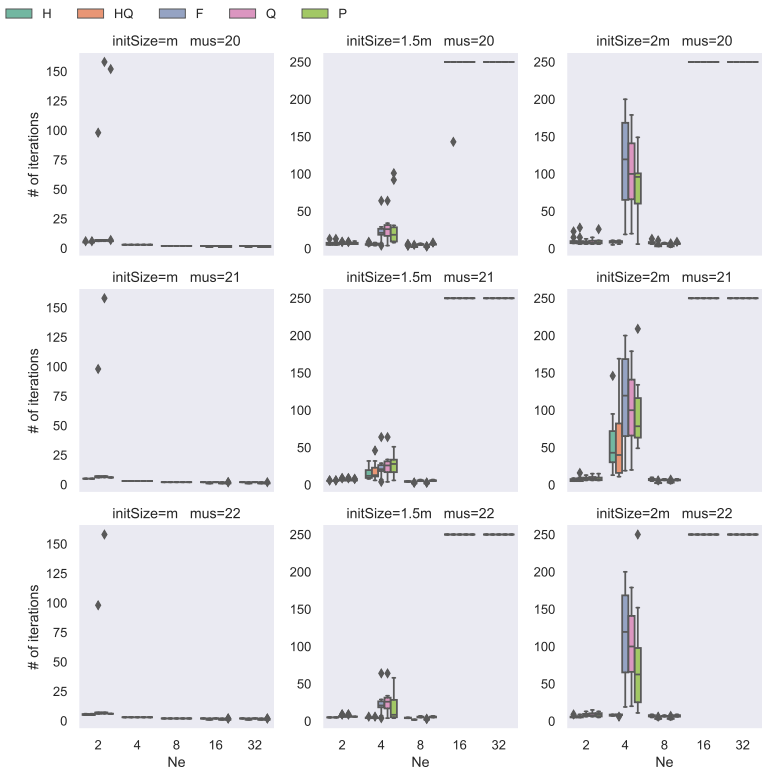
a) Number of iterations



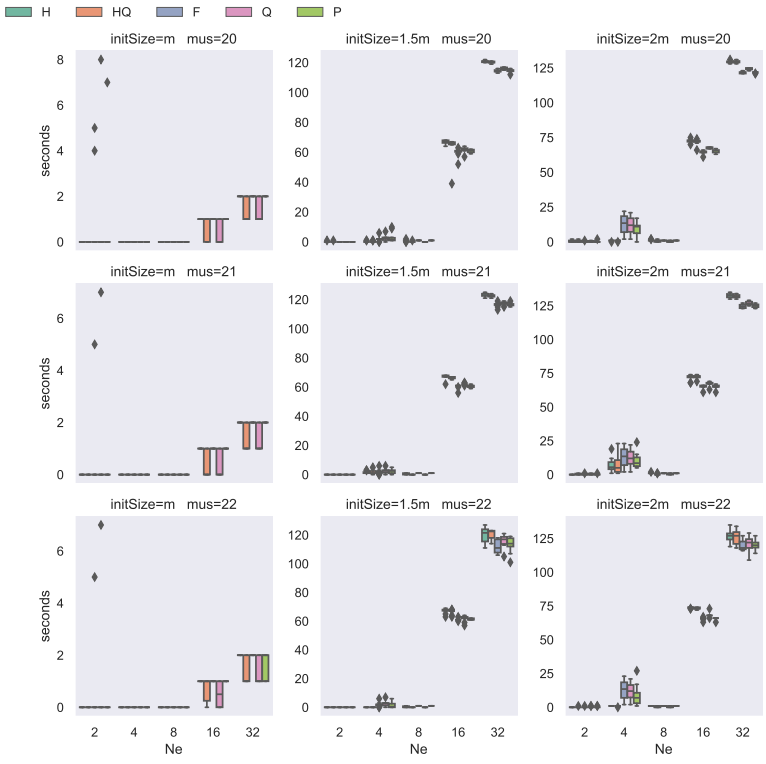
b) Time in seconds

Figure 4.10: Test results of QFEAST for the synthetic data where the smallest eigenvalues are computed

Up to this point, only the smallest eigenvalues are computed; however, as mentioned, various application areas might also require the interior ones. Therefore, to analyze the effects of the proposed method for the cases where interior eigenvalues are subject to investigation an extra spectrum in which the [0,10] interval contains 225, the [20,22] interval contains 50, and the [30,40] interval contains 225 eigenvalues created and a corresponding matrix constructed in a similar way as previously done. For this experiment, some slight changes are made. Firstly, the interval that algorithms have to search is provided as 19.99 and 22.01, the desired number of eigenvalues always is 50, and the tolerance for stopping is 10^{-12}. Moreover, a shift has to be applied in the inverse subspace algorithm because of the desired eigenvalues. 3 different shift values, 20, 21, and 22 are used for testing. Fig.(4.11) provides a comprehensive summary of the results. It should be noted that contrary to the previous setup, colors are used for the different methods, the x-axis of each subplot corresponds to the number of quadrature points and the global x-axis is for the shift values.



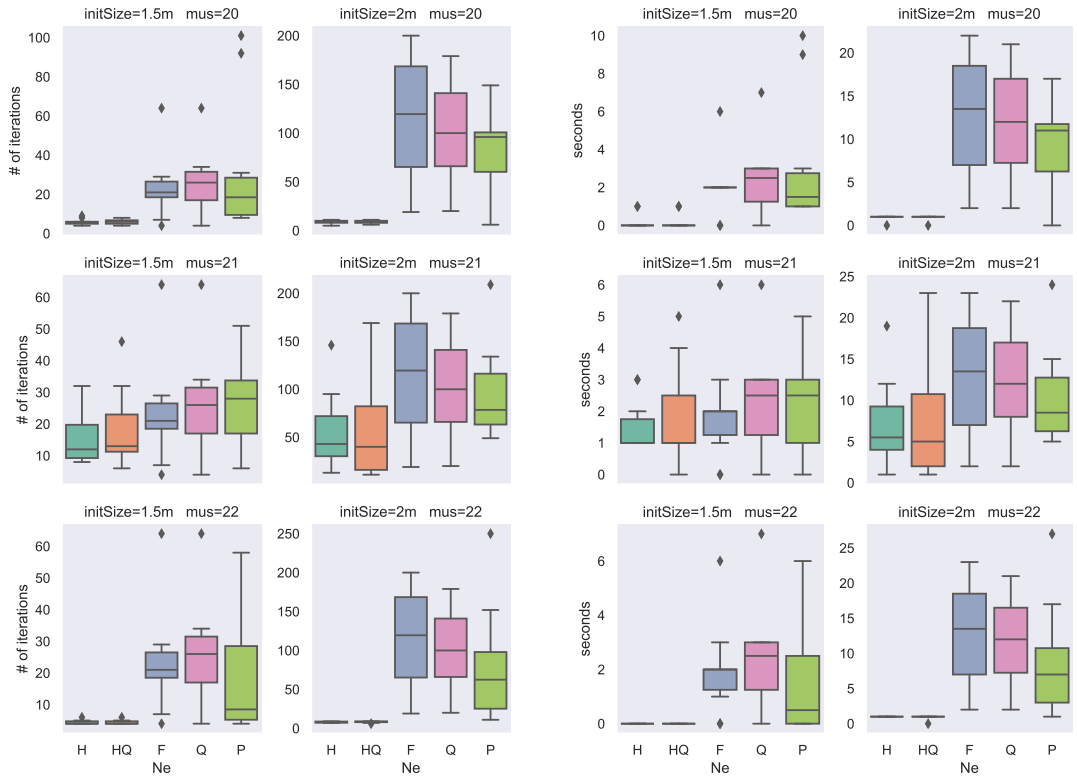
a) Number of iterations



b) Time in seconds

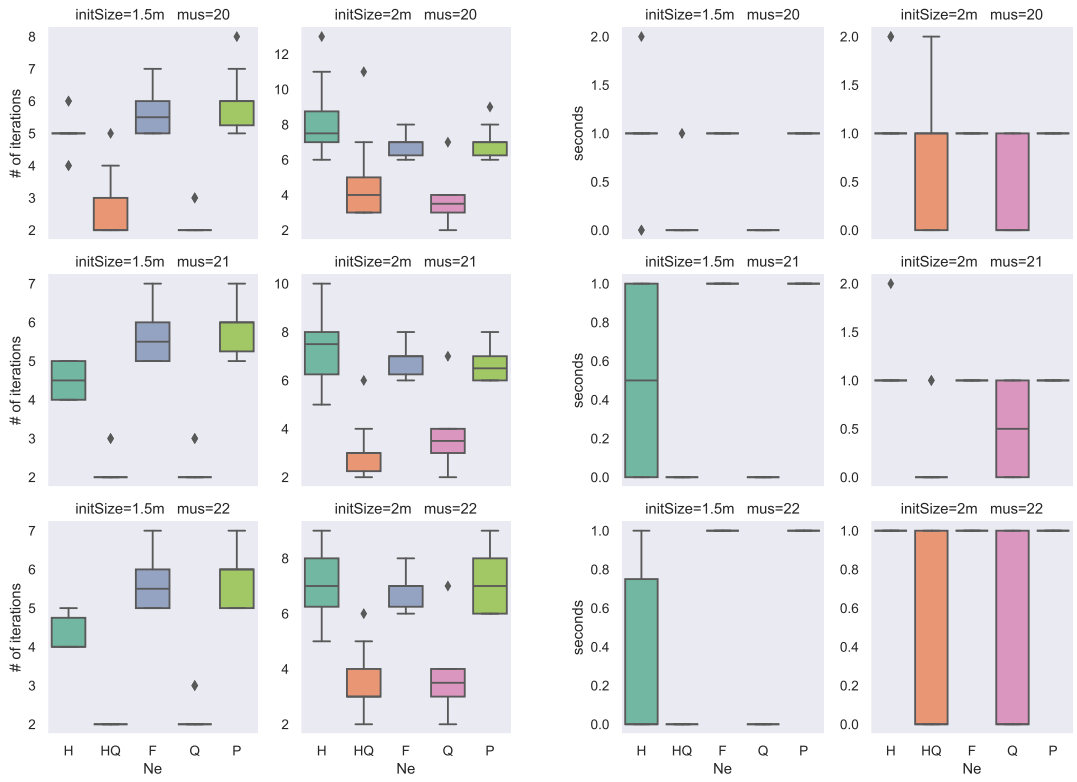
Figure 4.11: Test results of all methods for the synthetic data where the interior eigenvalues are computed

In the figure, when initSize is m there does not seem much difference and the results of quadrature numbers 2, 16, and 32 are similar, in fact for 16 and 32 methods do not converge; therefore, the results are summarized in Fig.(4.12) and Fig.(4.13). In contrast to previous spectrums, especially for Ne is 4, Hybrid approaches seem to dominate others and the effect of QR factorization becomes more important when the quadrature number is increased to 8.



a) Number of iterations b) Time in seconds

Figure 4.12: Test results of all methods for the synthetic data where the interior eigenvalues are computed, Ne is 4 and initSize is 1.5m, 2m



a) Number of iterations b) Time in seconds

Figure 4.13: Test results of all methods for the synthetic data where the interior eigenvalues are computed, Ne is 8 and initSize is 1.5m, 2m

4.2 Real-life Data

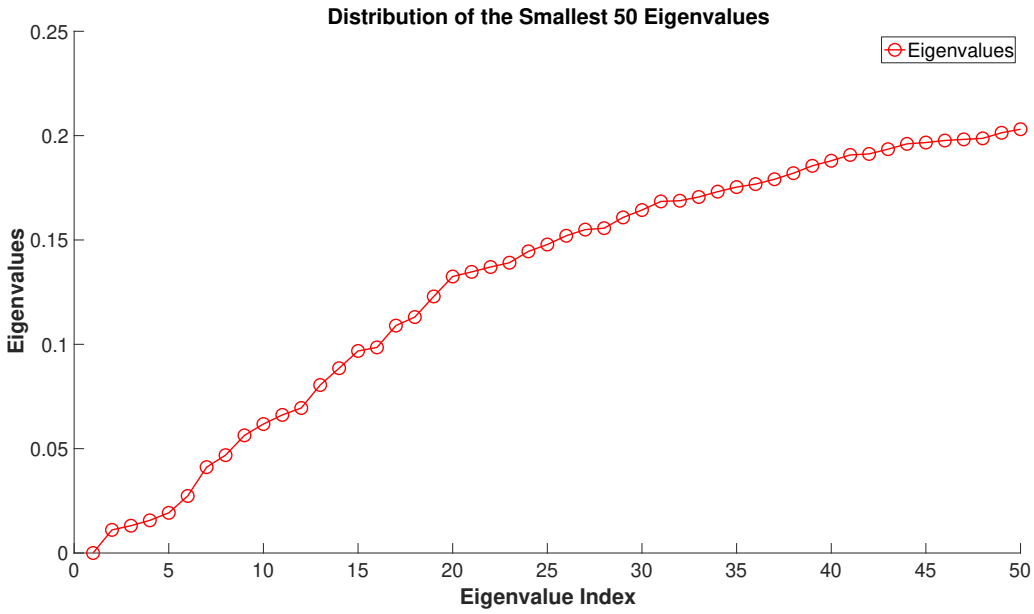
To test the methods for real-life data for Laplacian Graphs, Power Network Problems, Computational Fluid Dynamics, and Structural Problems matrices on the SuitSparse Matrix Collection [47] are used by each being real symmetric positive-definite and also sparse as mentioned. In contrast to synthetic data tests, instead of checking the residual error of the m^{th} smallest eigenvalue where m is the number of eigenvalues desired to compute, the residual error of all eigenvalues that are computed by the method is checked for convergence with the aim of investigating the effects of spurious eigenvalues. For all tests, the smallest eigenvalue is given as 0 and the largest one is given as the arithmetic mean of the m^{th} smallest and the $(m+1)^{th}$ smallest, which is not possible in real-life problems as the eigenvalues are not known apri-

orly; however, there are methods computing the number of eigenvalues lying inside an interval and domain experts could provide valuable guesses for the smallest eigenvalues. The desired number of eigenvalues is decided either based on the structure of the problem or the sharp increase in the distribution. Finally, to decide the stopping criteria, prior tests are conducted to detect where the residual error stagnates, caused by the dynamics of the problems, and the error bound is set. It should be kept in mind that even though these pieces of information are not available before the eigenpairs are computed, they are used for benchmarking the proposed methods, accordingly, experiments are conducted in the light of the problem's details.

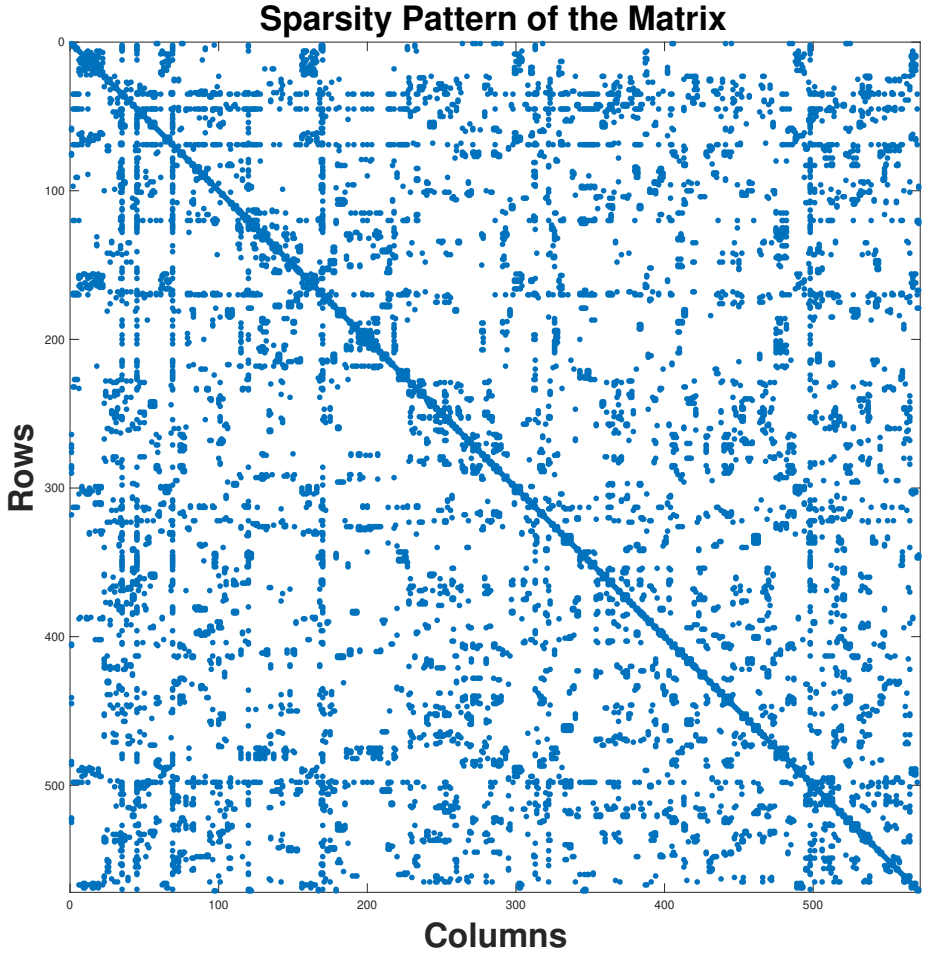
4.2.1 Spectral Clustering Problem

In Computer Sciences, clustering refers to the process in which data points (objects) are assigned to different groups with the purpose of objects labeled in the same group being more similar to each other than the ones in other groups [49]. Over the years, various clustering algorithms have been proposed one of which is Spectral Clustering. It is related to the Graph Theory and utilizes the Laplacian Graph, which is obtained by $L = D - A$ where D is the degree and A is the adjacency matrix of the graph. By construction, L has always at least one eigenvalue equal to zero, and the number of zero eigenvalues corresponds to the number of connected components of the Graph. Then, the second smallest eigenvalue of L, called the algebraic connectivity, measures how well-connected the graph is. Accordingly, eigenvectors of the smallest eigenvalues called the Fiedler vector, are used for clustering. Refer to [50] for a comprehensive analysis of Spectral Clustering and to [30] for parallel computation of the Fiedler vector.

In the thesis, the "micromass_10NN" matrix, which is an adjacency matrix, belonging to the "ML_Graph" group used in [51] is used. The matrix has a dimension of 571 with 9,668 non-zero elements. As described, the Laplacian matrix is obtained whose smallest 50 eigenvalues are given in Fig.(4.14) part a, also the part b of the same figure shows the sparsity pattern. Only the distribution of the smallest 50 eigenvalues is plotted as the desired number of eigenvalues is 20, which is the number of classes on the corresponding graph. The stopping criteria for the residual error is 10^{-15}.



a) Eigenvalues

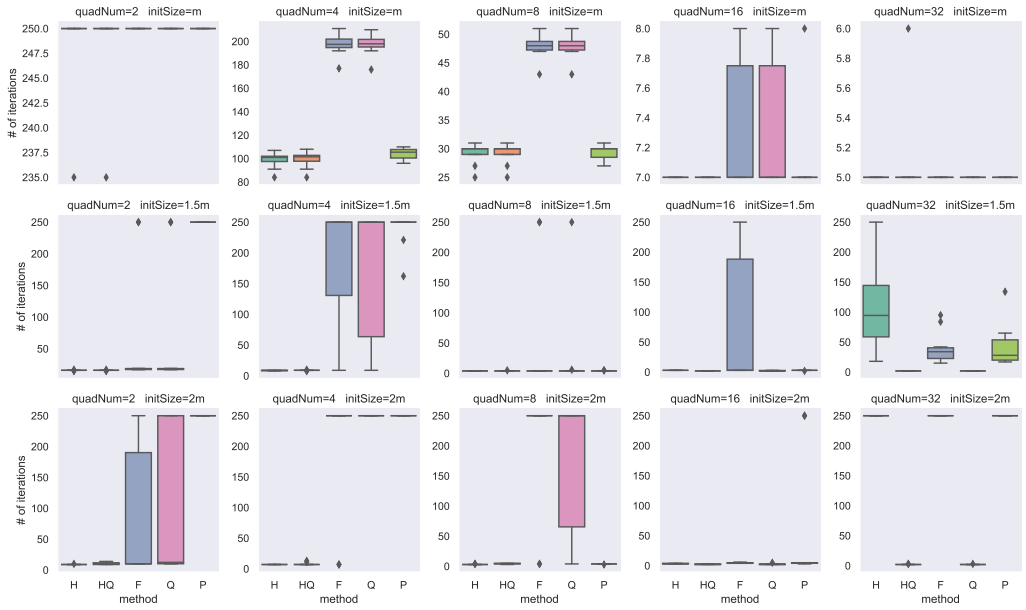


b) Sparsity

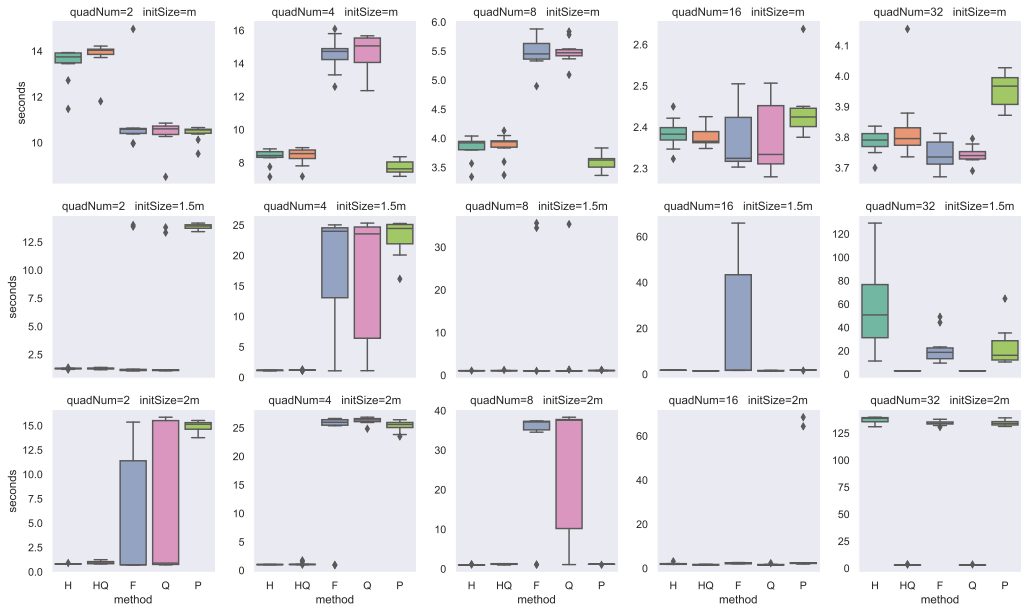
Figure 4.14: The smallest 50 eigenvalues and sparsity pattern of the Laplacian matrix

When the eigenvalue distribution of the Laplacian graph is investigated, it can be realized that it is neither similar to a sparse spectrum nor a discrete one. Instead, it resembles the dense one; accordingly, by transferring the gained knowledge from the dense case, it would be expected that Hybrid approaches will be superior. To determine the validity of the conclusion, we can analyze Fig.(4.15), parts a and b. In both parts, the global x-axis represents the variable Ne, and the global y-axis represents initSize. Within each subplot, the x-axis corresponds to the methods being compared. The key distinction between the two figures is the y-axis: in part a, it shows the number of iterations required, while in part b, it represents the time taken in seconds.

The first observation might be that for the smaller number of quadrature points, namely 2,4, and 8, except for quadNum is 2 and initSize is m, Hybrid approaches seem better than others in terms of the number of iterations they take and the time they require to compute. Moreover, for certain cases PFEAST dominates FEAST; whereas, the opposite is not true as there are no scenarios where FEAST is the obvious favorite against PFEAST. Furthermore, the addition of extra QR factorization does not have a clear decelerating effect, especially for the total duration, instead, it accelerates the process for the higher number of quadrature points. To finalize these figures, it is worth mentioning that the fastest step is where quadNum is 16 and initSize is m. Moreover, to conclude the discussion it is necessary to check the number of eigenvalues obtained by algorithms, which is given in Fig.(4.16). By discarding some exceptions, it could be concluded that the only problematic setup is when Ne and initSize are 32 and 2m, respectively. The reasoning for this could be that increasing dimension and quadrature numbers causes the floating point errors more pronounced. Nonetheless, the importance of the figure is that it shows QR factorization eliminates the appearance of spurious eigenvalues.



a) Number of iterations



b) Time in seconds

Figure 4.15: Iterations and time (seconds) for all methods for the Laplacian graph

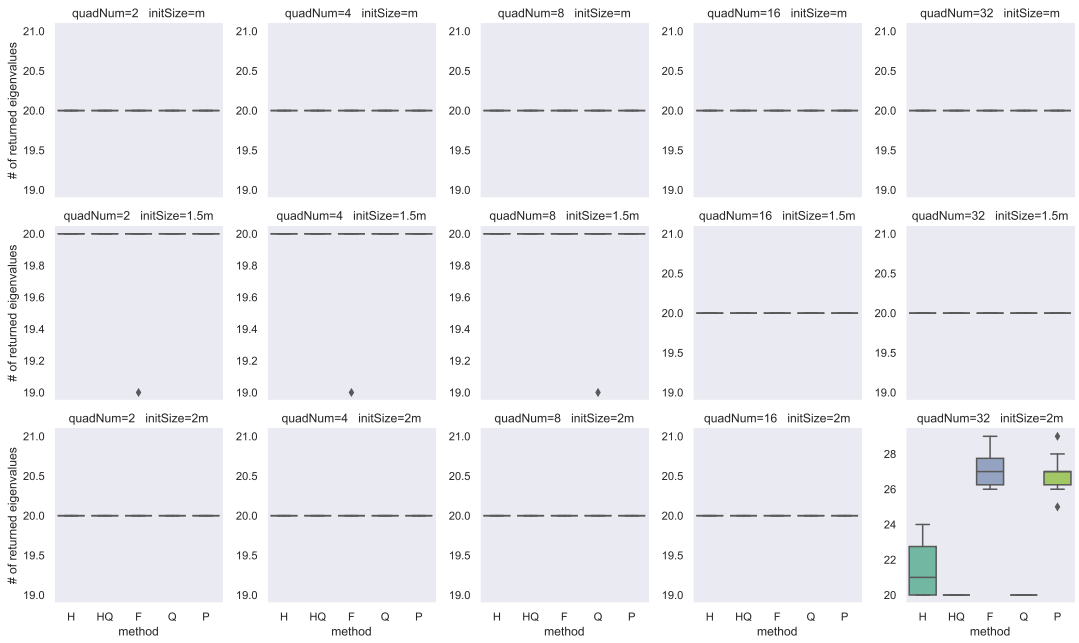
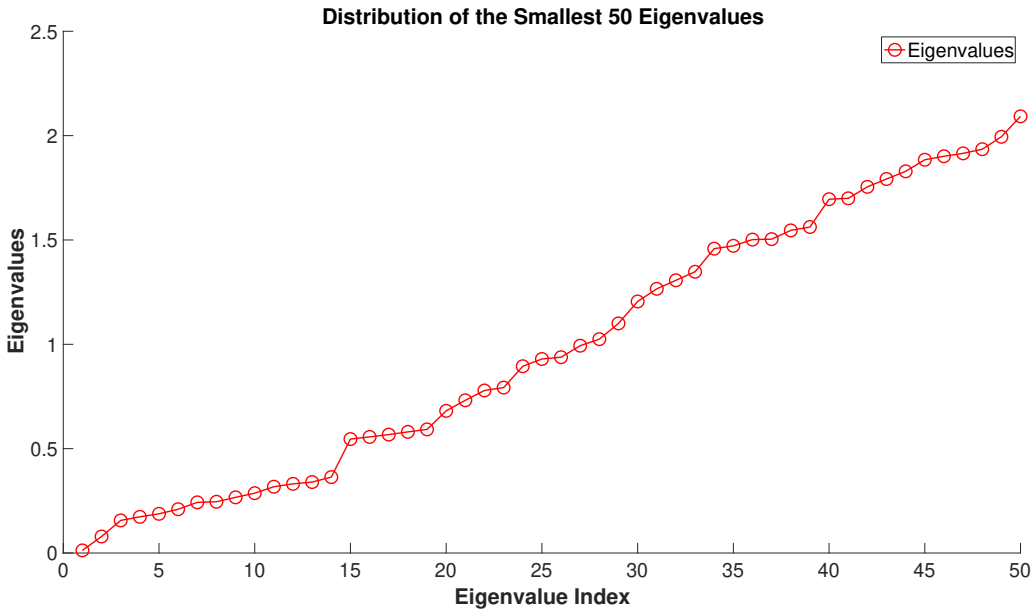


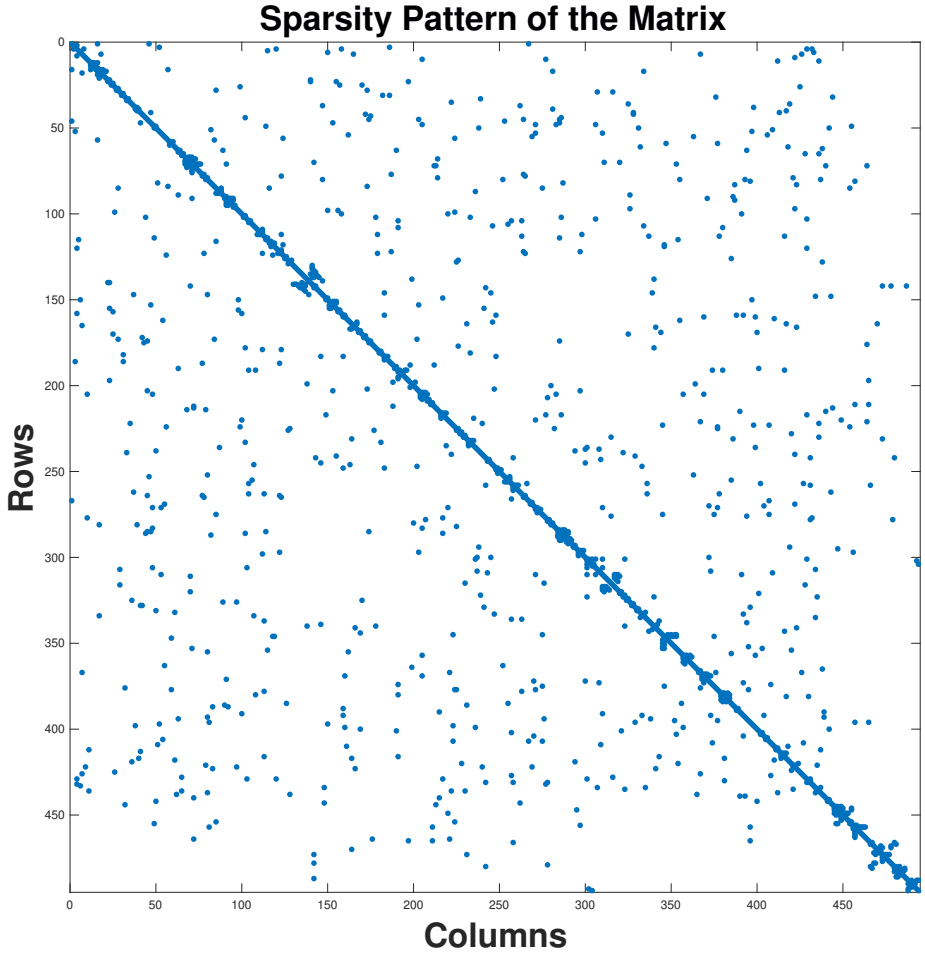
Figure 4.16: Number of eigenvalues obtained for all methods for the Laplacian graph

4.2.2 Power Network Problems

In power networks(systems), the Bus Admittance Matrix, which explains the topology of the system, is an $n \times n$ matrix representing the nodal admittances of the n buses. In realistic systems, as each bus is only connected to a few buses, the matrix tends to be quite sparse as it is in our case. The smallest eigenvalues of the matrix could be useful for various things like stability analysis. For the thesis, the "494_bus" matrix from the "HB" collection is used. The matrix has a dimension of 494 with 1,666 nonzero elements, and the distribution of the 50 smallest eigenvalues is given in Fig.(4.17) part a, also the sparsity pattern is shown in part b of the same figure.



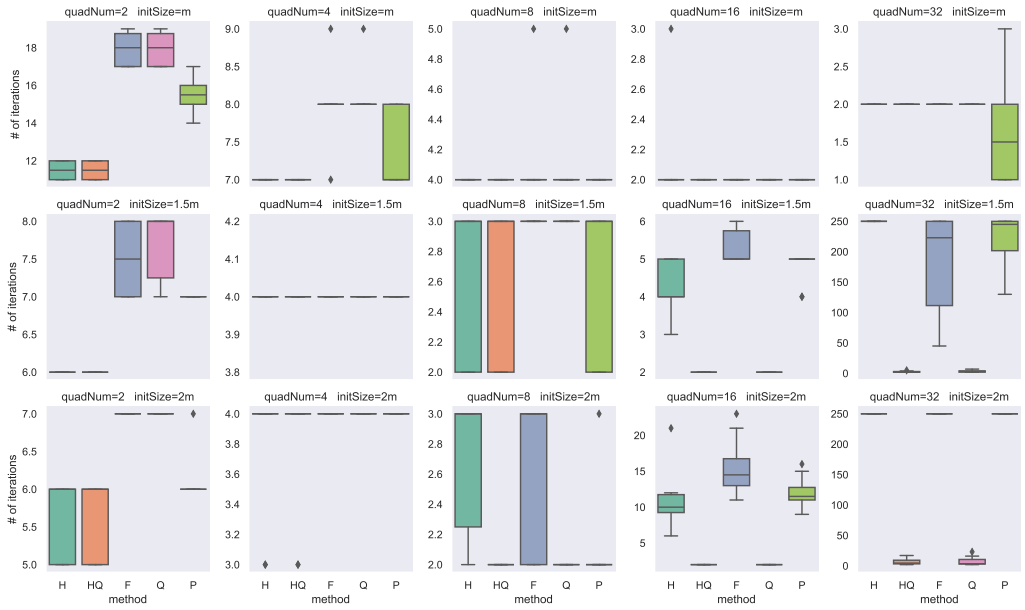
a) Eigenvalues



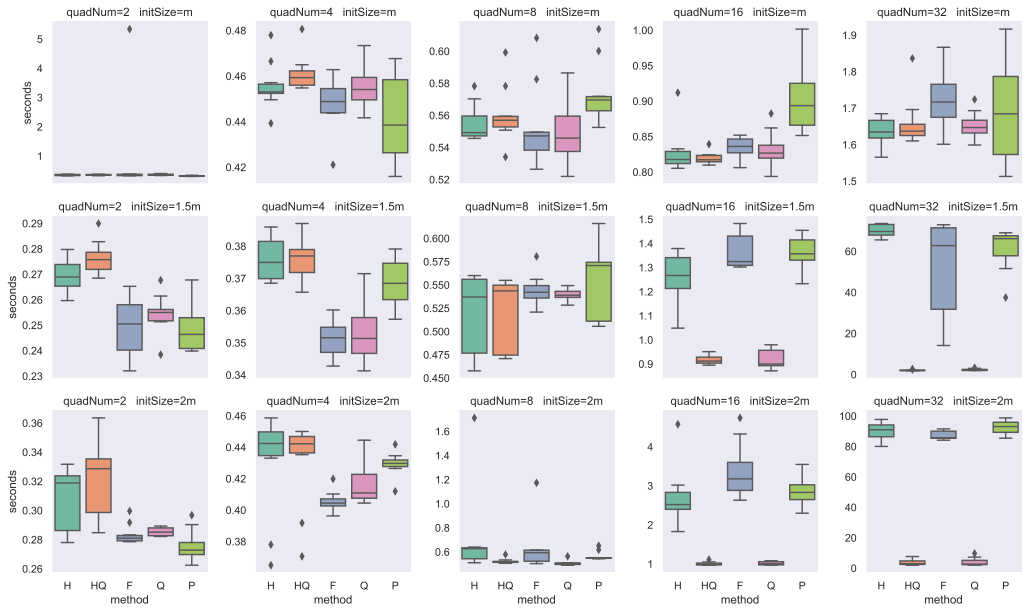
b) Sparsity

Figure 4.17: The smallest 50 eigenvalues and sparsity pattern of the Power Network matrix

In the tests, the smallest 14 eigenvalues are desired with the residual error of 10^{-12}, and the search interval is given as it is done for the previous test. In Fig.(4.18), in part a, the required iteration counts for algorithms to converge, and in part b, their running time can be seen. Accordingly, for the first three Ne's, there is not much difference but Hybrid approaches are never the worst in terms of the number of iterations; however, due to their per-iteration computational costs, for the total duration, other approaches are slightly better. Also, similar to the previous test, there is no favorite method when comparing FEAST and PFEAST, and for the higher number of quadrature points, QR approaches seem dominating; moreover, they are, especially QFEAST, never significantly behind FEAST. However, in contrast to the previous case, the fastest setup is where quadNum and initSize are 2 and 1.5m respectively. Followingly, when the number of eigenvalues obtained is checked in Fig.(4.19), it could be concluded that the only problematic setup is where Ne equals 32. A closer look reveals that even on it, QR factorization approaches returned exactly the desired number of eigenvalues, which seems to confirm one of the claims of the thesis. Moreover, it could be seen that the PFEAST approach is more stable in terms of the number of eigenvalues obtained.



a) Number of iterations



b) Time in seconds

Figure 4.18: Iterations and time (seconds) for all methods for the Power Network Problem

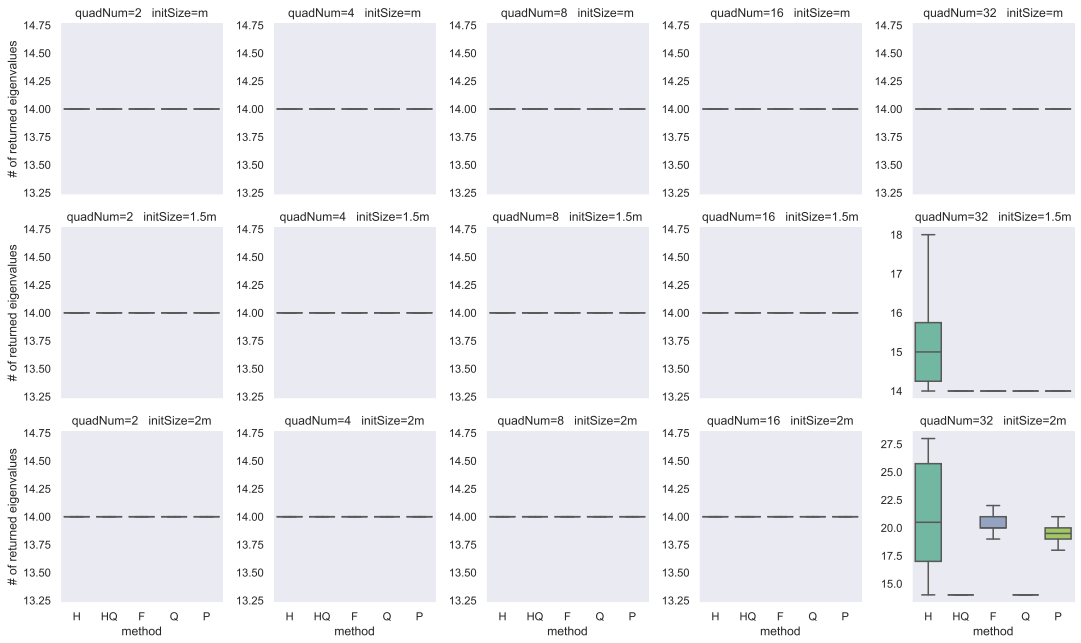
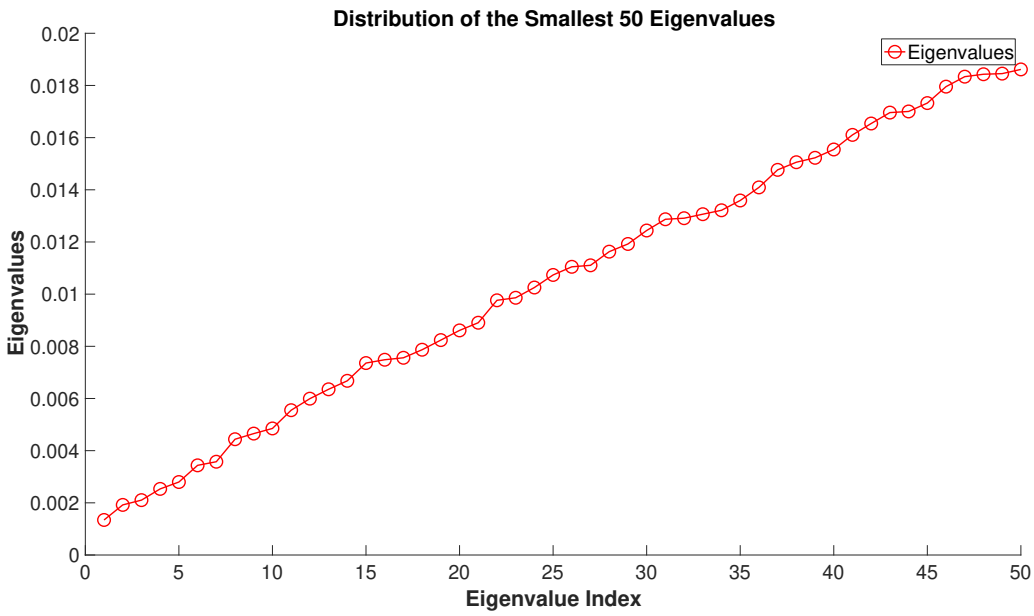


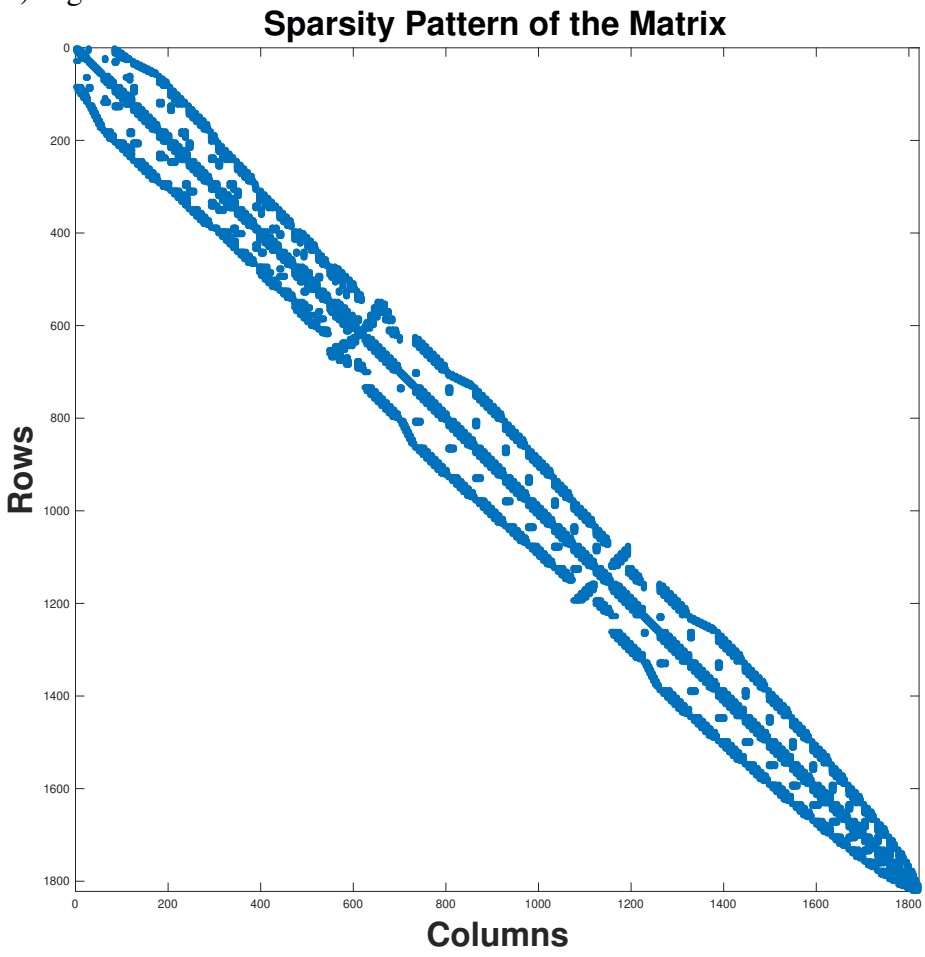
Figure 4.19: Number of eigenvalues obtained for all methods for the Power Network Problem

4.2.3 Computational Fluid Dynamics

Computational fluid dynamics (CFD), is the study of utilizing computers to forecast liquid and gas flows using the conservation of mass, momentum, and energy governing equations. For various reasons like improving the convergence of required iterative solvers the smallest eigenvalues of a matrix might be needed in CFD applications. The matrix used in testing is "ex3" named "2D Flow Past a Cylinder in Freestream" belonging to the FIDAP group. The dimension of the matrix is 1,821 with 52,685 nonzeros, and the distribution of the 50 smallest eigenvalues is given in part a of Fig.(4.20) which shows the sparsity pattern in part b.



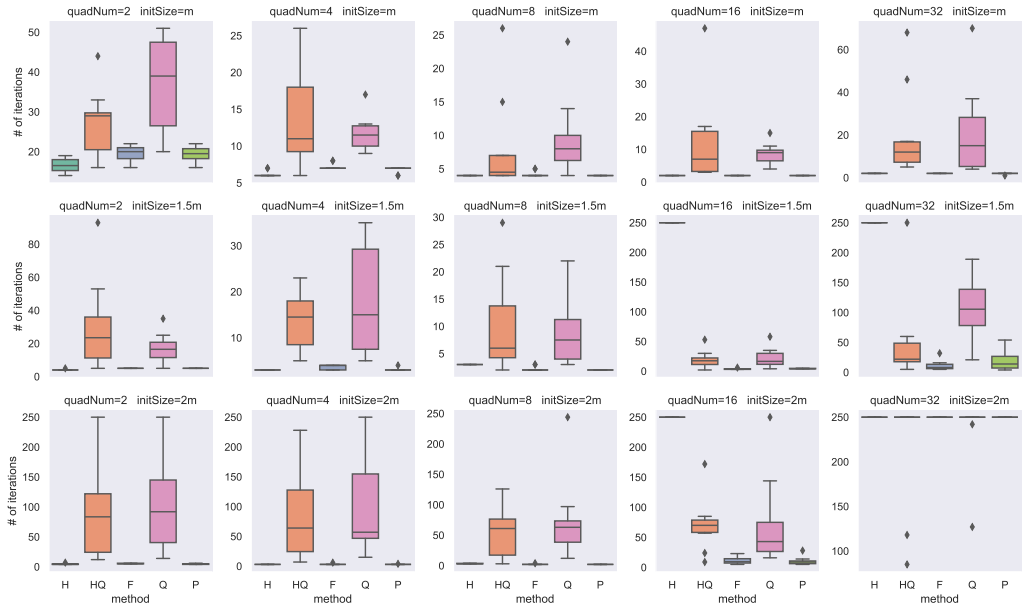
a) Eigenvalues



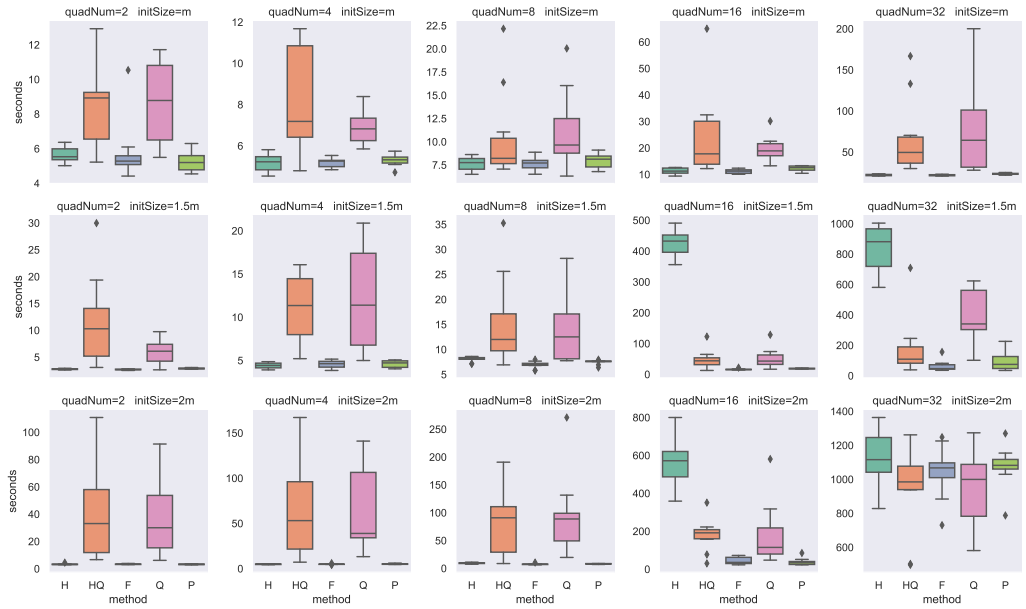
b) Sparsity

Figure 4.20: The smallest 50 eigenvalues and sparsity pattern of the CFD matrix

In the tests, the smallest 10 eigenvalues are desired with the residual error of 10^{-9}, and the search interval is given as it is done for the previous tests. In Fig.(4.21), iteration counts of algorithms and their running times are given, which depicts an interesting case in which almost in all setups algorithm with QR are the worst. That might be reasoned by the fact the CFD has the most resembling distribution to the dense one, in which QR approaches are worse. Moreover, again, none of FEAST and PFEAST dominate each other, further, for the smaller number of quadrature points, HFEAST is compatible with them. Further, for higher Ne's Hybrid approaches also do not seem preferable. Finally, these are the tests having the most total computation time due to the dimension of the matrix, and the fastest setups are for initSize is m, and quadNum is 2 or 4. Followingly, the investigation of the number of eigenvalues obtained in Fig.(4.21) would lead to similar conclusions to previous ones.



a) Number of iterations



b) Time in seconds

Figure 4.21: Iterations and time (seconds) for all methods for the CFD Problem

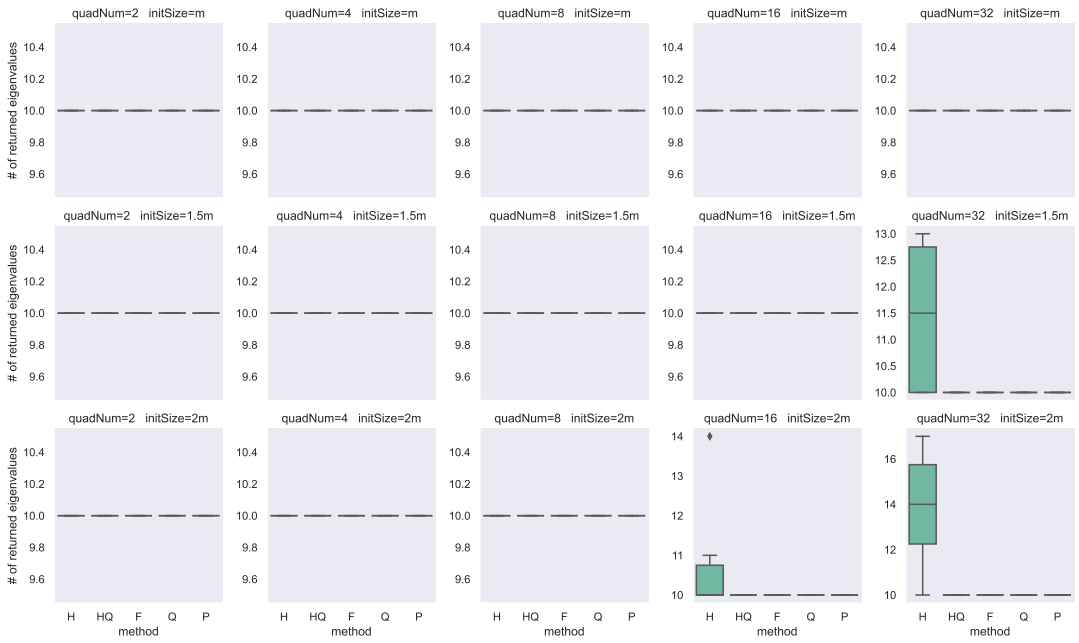
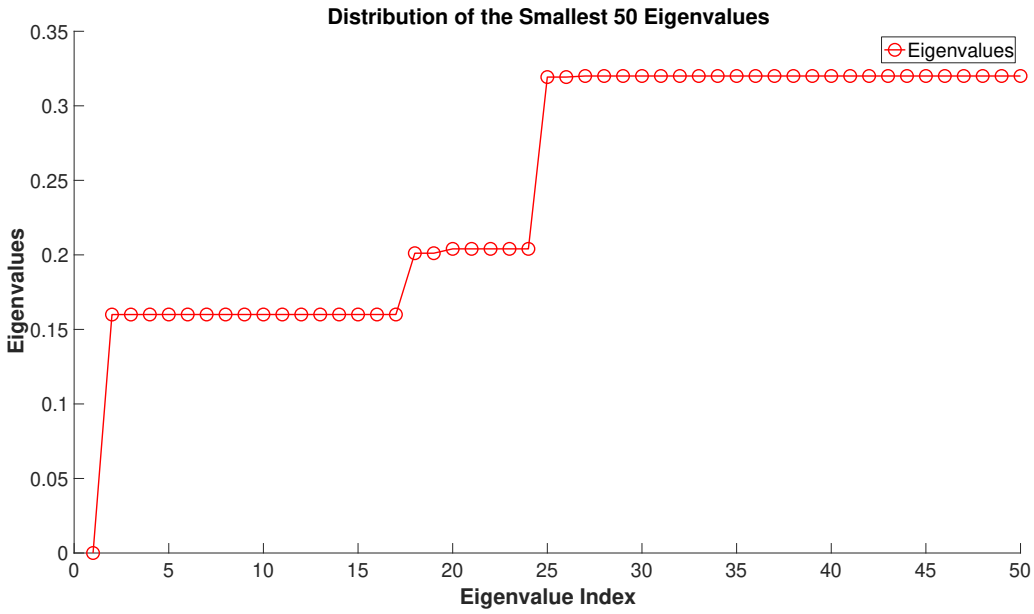


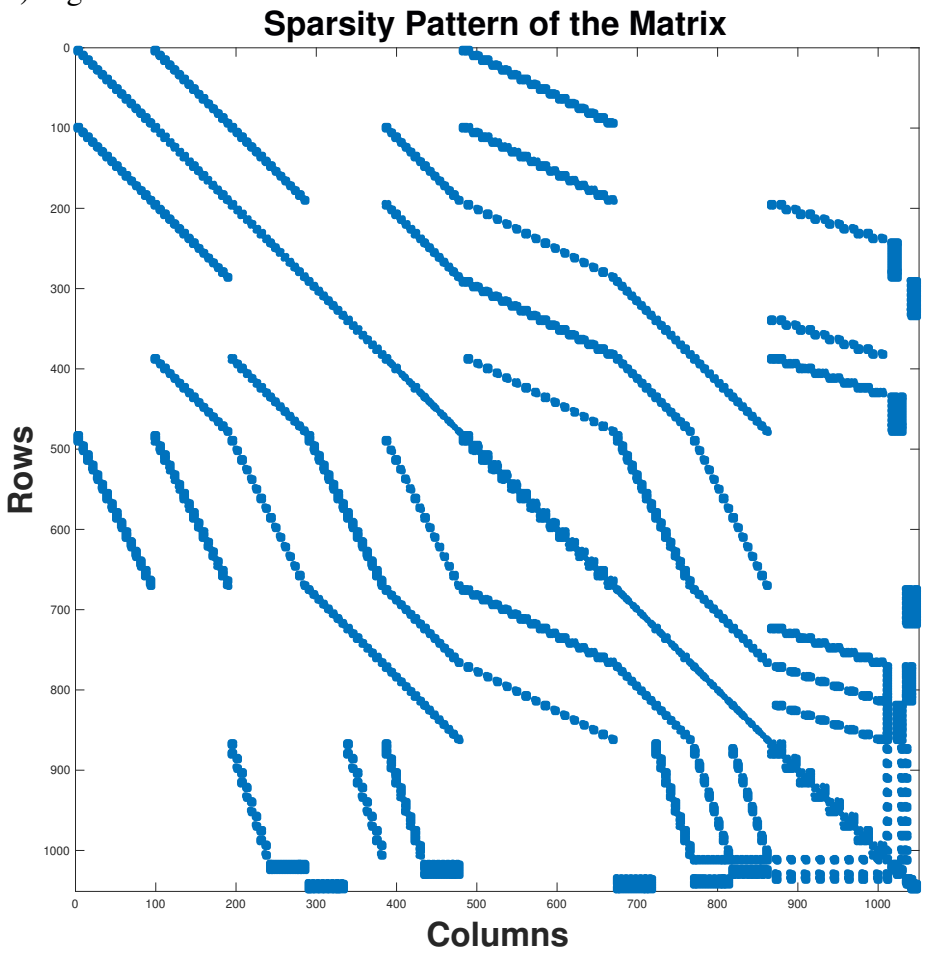
Figure 4.22: Number of eigenvalues obtained for all methods for the CFD Problem

4.2.4 Structural Engineering Problems

In structural engineering, the smallest eigenvalues of a matrix can be important for various analyses, such as Finite Element Model Verification, Dynamic Analysis, and Stiffness Matrix Analysis. These eigenvalues could be utilized to understand the natural frequencies and potential modes of vibration of a structure, which are essential for ensuring structural stability and performance. To test the effects of methods on this subject, the "msc01050" matrix from the "Boeing" collection is used. The matrix has a dimension of 1,050 with 26,198 non-zero elements, and the distribution of the 50 smallest eigenvalues is given in Fig.(4.23) part a. The differentiating thing for this interval is that it contains recurring eigenvalues. Also, the sparsity pattern of the matrix is given in the b part of the same Figure.



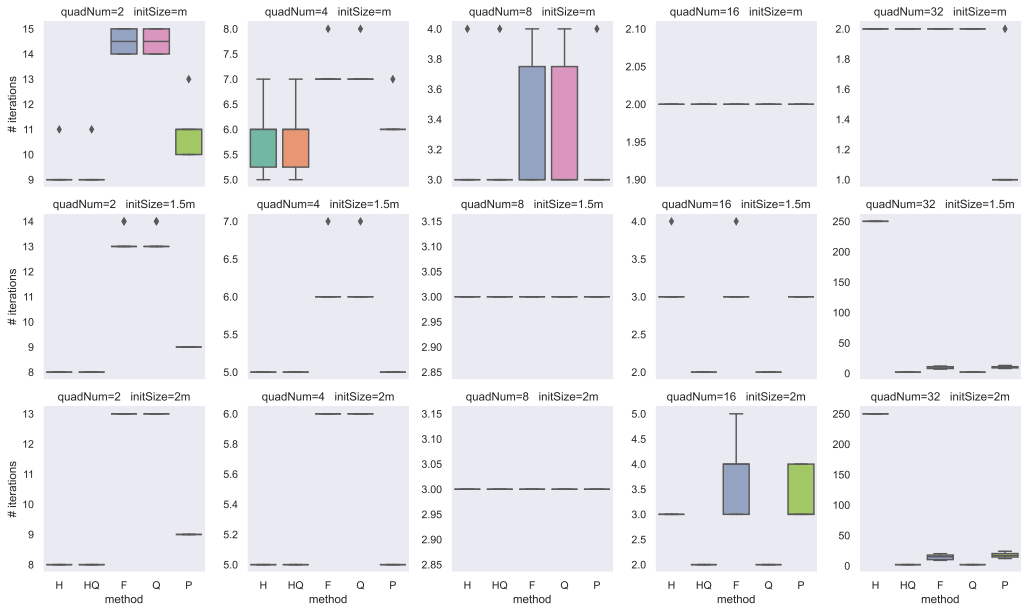
a) Eigenvalues



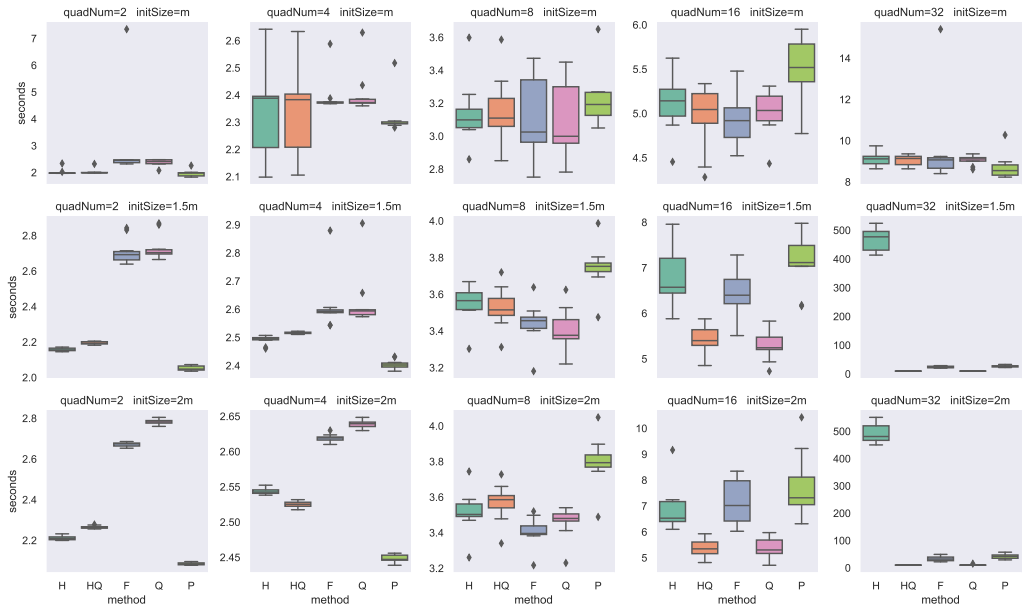
b) Sparsity

Figure 4.23: The smallest 50 eigenvalues and sparsity pattern of the Structural Engineering Problem matrix

In the tests, the smallest 24 eigenvalues are desired with the residual error of 10^{-9}, and the search interval is given as it is done for the previous tests. The results of the methods can be viewed in Fig.(4.24). Compared to the previous ones, this setup is the one where the least differences across methods are seen. A deeper investigation would reveal that for the smaller Ne's Hybrid approaches are preferable; however, for the higher ones they do not produce good results. Finally, the number of eigenvalues obtained, which are provided in Fig.(4.25), behave very similarly to the previous ones.



a) Number of iterations



b) Time in seconds

Figure 4.24: Iterations and time (seconds) for all methods for the Structural Engineering Problem

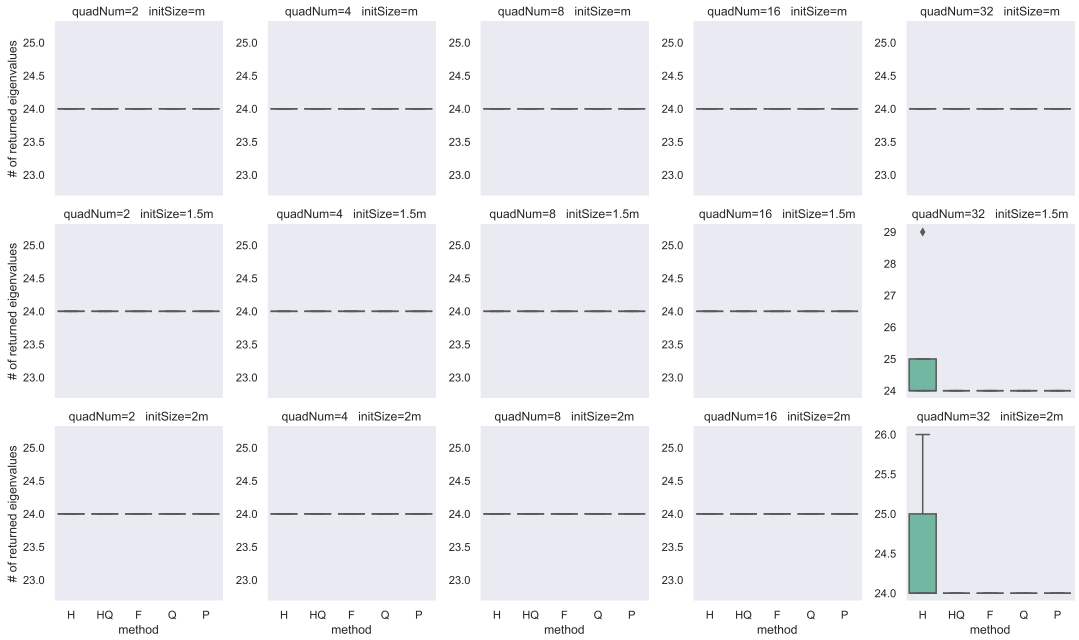


Figure 4.25: Number of eigenvalues obtained for all methods for the Structural Engineering Problem

CHAPTER 5

CONCLUSIONS

This thesis aims to investigate the convergence of FEAST eigensolver and the already proposed improvement over it that suggests adding a QR factorization to each step, across various eigenvalue spectrums. Furthermore, novel approaches are proposed to enhance the algorithm, one of which is adding the inverse subspace iteration (ISI) as a preprocessor to provide a better initial guess. The other one called the Hybrid approach, ensembles FEAST and ISI by iterating them alternatingly, which is also extended by either performing QR factorization or not. Their performances are also studied over various spectrums as well as compared with vanilla and QR FEAST.

Accordingly, to compare the methods both synthetic and real-life datasets are used. Synthetic eigenvalue spectrums are created under 3 categories: dense, discrete, and sparse. Real-life datasets are taken from various domains: Laplacian Graphs, Power Networks, Computational Fluid Dynamics, and Structural Engineering Problems. Based on the numerical experiments it is not possible to label a method as superior because their performance depends on the spectrum. In the dense spectrum, Hybrid approaches seem to be better in terms of the number of iterations they require to converge; however, when the total computation time is analyzed it is seen that as the number of quadrature points increases, they become slower than vanilla FEAST. After mentioning that, experiments indicate that there is no best number of quadrature points in terms of the total computation time as their performances change based on the spectrum type and matrix size. In other spectrum types, discrete and sparse, especially if the requested eigenvalue distribution includes a gap, QR ones are preferable as they generally decrease both the number of iterations and the duration proportionally to the width of the gap; nevertheless, even if they do not speed up the process,

the extra cost caused by the modification they have is negligible for the total duration, but it should be kept in mind that this is related to the matrix dimension. Besides, selecting the best among FEAST and PFEAST is not possible by analyzing the tests. Overall, if the eigenvalue distribution is known apriori, the better option could be decided; however, if it is unknown, choosing one of the improvements by venturing the extra computational cost is likely to pay off by decreasing the number of iterations hence, in general, the total computation time. Besides the number of iterations, the algorithms are evaluated for the number of eigenvalues obtained. Based on the findings, the spurious eigenvalues do not appear for the smaller number of quadrature points and initial sizes but this is not the case for the higher number of quadrature points for which spurious eigenvalues are detected. Nevertheless, even for the latter, QR factorization approaches find exactly the requested number of eigenvalues.

As future work, the proposed methods could be enabled for non-symmetric/non-real matrices. Also, testing for higher dimensional matrices and quadrature numbers might be useful to better talk about the performances. The theoretical analysis of the proposed methods is the crucial step for the comparison to detect the best option. Additionally, the experiments suggest a relationship between the eigenvalue spectrum and the performance of the methods. Different modifications tend to perform better on varying distributions, highlighting the potential of this connection. Identifying the optimal setup for specific spectra will be an important focus for future work. Besides, as mentioned previously, parallelizing the proposed methods seems possible.

REFERENCES

[1] B. Ghojogh, F. Karray, and M. Crowley, "Eigenvalue and generalized eigenvalue problems: Tutorial," 2023.

[2] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, vol. 662. Oxford Clarendon, 1965.

[3] I. Jolliffe, *Principal Component Analysis*. Springer, 2011.

[4] G. Peters and J. H. Wilkinson, "$Ax = \lambda Bx$ and the generalized eigenproblem," *SIAM Journal on Numerical Analysis*, vol. 7, no. 4, pp. 479–492, 1970.

[5] B. N. Parlett, *The Symmetric Eigenvalue Problem*, vol. 20 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics, 1998.

[6] S. Güttel and F. Tisseur, "The nonlinear eigenvalue problem," *Acta Numerica*, vol. 26, p. 1–94, 2017.

[7] G. Strang, *Linear algebra and its applications*. Thomson, Brooks/Cole, 2006.

[8] B. P. Brooks, "The coefficients of the characteristic polynomial in terms of the eigenvalues and the elements of an n×n matrix," *Applied Mathematics Letters*, vol. 19, no. 6, pp. 511–515, 2006.

[9] E. Polizzi, "Density-matrix-based algorithm for solving eigenvalue problems," *Physical Review B*, Mar 2009.

[10] H. Rutishauser, "Solution of eigenvalue problems with the lr-transformation," *NBS Appl. Math. Series*, vol. 49, p. 47–81, 1958.

[11] J. G. F. Francis, "The qr transformation a unitary analogue to the lr transformation–part 1," *The Computer Journal*, vol. 4, 1961.

[12] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*. 2011.

[13] S. Andrilli and D. Hecker, "Chapter 9 - numerical techniques," in *Elementary Linear Algebra (Sixth Edition)* (S. Andrilli and D. Hecker, eds.), pp. 401–439, Academic Press, sixth edition ed., 2023.

[14] L. N. Trefethen and D. Bau, III *Numerical linear algebra*, Jan 1997.

[15] B. N. Parlett, "The rayleigh quotient iteration and some generalizations for non-normal matrices," *Mathematics of Computation*, vol. 28, pp. 679–693, 1974.

[16] B. N. Parlett and W. G. Poole, Jr., "A geometric theory for the qr, lu and power iterations," *SIAM Journal on Numerical Analysis*, vol. 10, p. 389–412, Apr 1973.

[17] Z. Jia and G. W. Stewart, "An analysis of the rayleigh–ritz method for approximating eigenspaces," *Mathematics of Computation*, vol. 70, 2000.

[18] Y. Saad, "The origin and development of krylov subspace methods," *Computing in Science & Engineering*, vol. 24, p. 28–39, Jul 2022.

[19] W. E. Arnoldi, "The principle of minimized iterations in the solution of the matrix eigenvalue problem," *Quarterly of Applied Mathematics*, vol. 9, no. 1, p. 17–29, 1951.

[20] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *Journal of Research of the National Bureau of Standards*, vol. 45, p. 255, Oct 1950.

[21] M. Bellalij, G. Meurant, and H. Sadok, "The distance of an eigenvector to a krylov subspace and the convergence of the arnoldi method for eigenvalue problems," *Linear Algebra and its Applications*, vol. 504, pp. 387–405, 2016.

[22] Z. Jia, "The convergence of generalized lanczos methods for large unsymmetric eigenproblems," *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 3, pp. 843–862, 1995.

[23] Z. Teng and R.-C. Li, "Convergence analysis of lanczos-type methods for the linear response eigenvalue problem," *Journal of Computational and Applied Mathematics*, vol. 247, pp. 17–33, 2013.

[24] Y. Saad, "Variations on arnoldi's method for computing eigenelements of large unsymmetric matrices," *Linear Algebra and its Applications*, vol. 34, pp. 269–295, 1980.

[25] M. BELLALIJ, Y. SAAD, and H. SADOK, "Further analysis of the arnoldi process for eigenvalue problems," *SIAM Journal on Numerical Analysis*, vol. 48, no. 2, pp. 393–407, 2010.

[26] A. H. Sameh and J. A. Wisniewski, "A trace minimization algorithm for the generalized eigenvalue problem," *SIAM Journal on Numerical Analysis*, vol. 19, no. 6, pp. 1243–1259, 1982.

[27] A. Sameh and Z. Tong, "The trace minimization method for the symmetric generalized eigenvalue problem," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1, pp. 155–175, 2000. Numerical Analysis 2000. Vol. III: Linear Algebra.

[28] A. M. Klinvex, "Parallel symmetric eigenvalue problem solvers," 2015.

[29] A. Klinvex, F. Saied, and A. Sameh, "Parallel implementations of the trace minimization scheme tracemin for the sparse symmetric eigenvalue problem," *Computers & Mathematics with Applications*, vol. 65, no. 3, pp. 460–468, 2013. Efficient Numerical Methods for Scientific Applications.

[30] M. Manguoglu, E. Cox, F. Saied, and A. Sameh, "Tracemin-fiedler: A parallel algorithm for computing the fiedler vector," pp. 449–455, 06 2010.

[31] A. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM Journal on Scientific Computing*, vol. 23, 11 2000.

[32] A. V. Knyazev, M. E. Argentati, I. Lashuk, and E. E. Ovtchinnikov, "Block locally optimal preconditioned eigenvalue xolvers (blopex) in hypre and petsc," *SIAM Journal on Scientific Computing*, vol. 29, p. 2224–2239, Jan 2007.

[33] P. T. Peter Tang and E. Polizzi, "Feast as a subspace iteration eigensolver accelerated by approximate spectral projection," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 2, pp. 354–390, 2014.

[34] E. Di Napoli, E. Polizzi, and Y. Saad, "Efficient estimation of eigenvalue counts in an interval," *Numerical Linear Algebra with Applications*, vol. 23, p. 674–692, Mar 2016.

[35] N. J. Higham *Functions of matrices*, Jan 2008.

[36] B. Gavin and E. Polizzi, "Enhancing the performance and robustness of the feast eigensolver," *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep 2016.

[37] L. Krämer, E. Di Napoli, M. Galgon, B. Lang, and P. Bientinesi, "Dissecting the feast algorithm for generalized eigenproblems," *Journal of Computational and Applied Mathematics*, vol. 244, p. 1–9, May 2013.

[38] M. Galgon, L. Krämer, B. Lang, A. Alvermann, H. Fehske, and A. Pieper, "Improving robustness of the feast algorithm and solving eigenvalue problems from graphene nanoribbons," *PAMM*, vol. 14, p. 821–822, Dec 2014.

[39] S. Kajpust, "Variations of the feast eigenvalue algorithm," 2014.

[40] P. T. P. Tang, J. Kestyn, and E. Polizzi, "A new highly parallel non-hermitian eigensolver," 2014.

[41] S. Güttel, E. Polizzi, P. T. P. Tang, and G. Viaud, "Zolotarev quadrature rules and load balancing for the feast eigensolver," *SIAM Journal on Scientific Computing*, vol. 37, no. 4, pp. A2100–A2122, 2015.

[42] Y. P. Hong and C.-T. Pan, "Rank-revealing qr factorizations and the singular value decomposition," *Mathematics of Computation*, vol. 58, p. 213, Jan 1992.

[43] B. E. Gavin, "Inexact and nonlinear extensions of the feast eigenvalue algorithm," no. 1341, 2018. Available at: `https://scholarworks.umass.edu/dissertations_2/1341`.

[44] M. Rizwan, E. Jung, J. Choi, *et al.*, "Revisiting the performance optimization of qr factorization on intel knl and skl multiprocessors," *Journal of Supercomputing*, vol. 80, pp. 13813–13836, 2024.

[45] M. Anderson, G. Ballard, J. Demmel, and K. Keutzer, "Communication-avoiding qr decomposition for gpus," in *2011 IEEE International Parallel & Distributed Processing Symposium*, pp. 48–58, 2011.

[46] Y. Saad, *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2003.

[47] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Transactions on Mathematical Software*, vol. 38, pp. 1:1–1:25, December 2011.

[48] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[49] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, p. 264–323, sep 1999.

[50] U. von Luxburg, "A tutorial on spectral clustering," *CoRR*, vol. abs/0711.0189, 2007.

[51] D. Pasadakis, C. L. Alappat, O. Schenk, and G. Wellein, "K-way p-spectral clustering on grassmann manifolds," 2020.