THE E^2 BATHE SUBSPACE ITERATION METHOD

by

Bryce Daniel Wilkins

B.S., United States Military Academy (2017)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Signature redacted

Author ..... Department of Mechanical Engineering
May 22, 2019

Signature redacted

Certified by ..... Klaus-Jürgen Bathe
Professor of Mechanical Engineering
Thesis Supervisor

Signature redacted

Certified by ..... Mavis Driscoll
Principal Member of the Technical Staff, Draper
Thesis Supervisor

Signature redacted

Accepted by ..... Nicolas Hadjiconstantinou
Chairman, Department Committee on Graduate Theses

THE E^2 BATHE SUBSPACE ITERATION METHOD

by

Bryce Daniel Wilkins

Submitted to the Department of Mechanical Engineering
on May 28, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Since its development in 1971, the Bathe subspace iteration method has been widely-used to solve the generalized symmetric-definite eigenvalue problem. The method is particularly useful for solving large eigenvalue problems when only a few of the least dominant eigenpairs are sought. In reference [18], an enriched subspace iteration method was proposed that accelerated the convergence of the basic method by replacing some of the iteration vectors with more effective turning vectors. In this thesis, we build upon this recent acceleration effort and further enrich the subspace of each iteration by replacing additional iteration vectors with our new turning-of-turning vectors.

We begin by reviewing the underpinnings of the subspace iteration methodology. Then, we present the steps of our new algorithm, which we refer to as the Enriched-Enriched (E^2) Bathe subspace iteration method. This is followed by a tabulation of the number of floating point operations incurred during a general iteration of the E^2 algorithm. Additionally, we perform a simplified convergence analysis showing that the E^2 method converges asymptotically at a faster rate than the enriched method. Finally, we examine the results from several test problems that were used to illustrate the E^2 method and to assess its potential computational savings compared to the enriched method.

The sample results for the E^2 method are consistent with the theoretical asymptotic convergence rate that was obtained in our convergence analysis. Further, the results from the CPU time tests suggest that the E^2 method can often provide a useful reduction in computational effort compared to the enriched method, particularly when relatively few iteration vectors are used in comparison with the number of eigenpairs that are sought.

Thesis Supervisor: Klaus-Jürgen Bathe
Title: Professor of Mechanical Engineering

Thesis Supervisor: Mavis Driscoll
Title: Principal Member of the Technical Staff, Draper

Acknowledgments

I would like to thank my research advisor, Professor Klaus-Jürgen Bathe, for his patience and for guiding me through the completion of this thesis. I have learned a lot from this experience, and I am grateful for his investment in me over the last two years.

I would like to thank my Draper advisor, Dr. Mavis Driscoll, for her thoughtful mentorship and for giving me ample flexibility to pursue my academic interests. Our weekly meetings were great for exploring some very neat aspects of mathematics.

I would like to thank Draper for funding my graduate study at MIT, and I would also like to thank the U.S. Army for giving me this opportunity to pursue graduate education.

I would like to thank my roommate and West Point classmate, Adam Kratch, for two years of adventures in Boston. Additionally, I would like to thank the rest of my friends and family for their support and encouragement while I have been here.

I am also grateful to the several mentors that I have had over the last few years who have continued to help me become a better version of myself. These people include, but are not limited to, Dr. Theodore Hromadka, Mrs. Laura Hromadka, Colonel Joseph Lindquist, Captain Brian Drapp, Major Jonathan Paynter, Major John Morrow, Dr. Christopher Morey, and Mrs. Jennifer Miller.

Finally, I would like to thank my Mom for, among many things, (almost) always answering my phone calls!

This thesis is dedicated to my family.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

Chapter 1

INTRODUCTION

In this thesis, we discuss a new extension of the Bathe subspace iteration method for solving the generalized eigenvalue problem:

$$\boldsymbol{K}\boldsymbol{\phi} = \omega^2 \boldsymbol{M}\boldsymbol{\phi}, \qquad \text{where } \boldsymbol{K} \in \mathbb{R}^{n\times n},\ \boldsymbol{M} \in \mathbb{R}^{n\times n}, \quad \boldsymbol{\phi} \in \mathbb{R}^n,\ \text{ and } \omega^2 \in \mathbb{R} \tag{1.1}$$

For simplicity, the matrices $\boldsymbol{K}$ and $\boldsymbol{M}$ in (1.1) are assumed to be symmetric and positive definite. However, as discussed in reference [5], the subspace iteration method is also applicable when $\boldsymbol{K}$ is indefinite, as well as when $\boldsymbol{M}$ is semi-definite.

Eigenvalue problems such as (1.1) are often encountered in the finite element analysis of structures when modal superposition is used to model the structural dynamics. In the context of finite element analysis, $\boldsymbol{K}$ and $\boldsymbol{M}$ correspond to the stiffness matrix and mass matrix, respectively, of the system. For structural problems in particular, the eigenvalues of (1.1) are interpreted physically as the natural vibration frequencies of the system, and the corresponding eigenvectors represent the modal shapes.

Many structures respond to loads primarily in just a few of the lowest frequencies and modes of the system [7]. When this is the case, it is possible to create accurate models of the structural dynamics without necessarily computing all n eigenpairs [8]. Instead, it is usually sufficient to compute only the first $p \ll n$ eigenpairs of the least dominant subspace of $\boldsymbol{K}$ and $\boldsymbol{M}$. These p eigenpairs of interest are written compactly

as

$$\boldsymbol{K}\boldsymbol{\Phi} = \boldsymbol{M}\boldsymbol{\Phi}\boldsymbol{\Omega}^2, \qquad \text{where } \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_p \end{bmatrix} \quad \boldsymbol{\Omega}^2 = \text{diag}(\omega_1^2, \omega_2^2, \ldots, \omega_p^2) \tag{1.2}$$

The generalized eigenvalues and eigenvectors of (1.2) are assumed to be ordered as follows:

$$\begin{aligned} &0 < \omega_1^2 \leq \omega_2^2 \leq \cdots \leq \omega_p^2 \leq \cdots \leq \omega_n^2 < \infty \\ &\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_p, \ldots, \boldsymbol{\phi}_n \end{aligned} \tag{1.3}$$

As noted in references [1, 18, 26], the solution of (1.2) is often the most time-consuming part of the dynamic response analysis of large structures. Consequently, there is great interest in developing new numerical schemes, or improving existing and reliable algorithms, for solving the generalized eigenvalue problem.

As indicated by the review conducted in reference [8], there are several numerical algorithms that can be used to solve the generalized eigenvalue problem at hand. When determining which one of the several existing algorithms to use, it is necessary to consider characteristics of the problem such as the order of the system, the mean half-bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$, the densities of $\boldsymbol{K}$ and $\boldsymbol{M}$, and also the number of eigenpairs that are required.

In this work, our focus is on the solution of eigenvalue problems where n is very large (say, n between 10^4 and 10^7), $\boldsymbol{K}$ and $\boldsymbol{M}$ are sparse, and relatively few of the least dominant eigenpairs are sought. When n is very large, transformation methods, such as the Householder-QR algorithm, and search-type methods, such as the determinant search approach, are not practical because of the excessive computational cost and memory resources that these methods require [13, 23]. Rather, more practical methods for the case when n is very large are the Lanczos method and the Bathe subspace iteration method. Both of these methods have consistently done well solving the large eigenvalue problems that are the focus of this thesis [19]. The Lanczos method has been the subject of much research, and the reader is referred to references [10, 12, 14, 15, **?**, 21, 22, 24, 27] for more details.

The subspace iteration method is a competitive alternative to the Lanczos method

and was introduced by Klaus-Jürgen Bathe in 1971 [2]. Similarly to the Lanczos method, the Bathe subspace iteration method is useful for the solution of large eigenvalue problems, particularly when relatively few of the least-dominant eigenpairs are sought. Several of the attractive properties of the Bathe subspace iteration method are as follows:

1. The algorithm is robust and efficient.

2. The algorithm converges quickly when the starting subspace is close to the target subspace. This is a desirable property because the eigenvectors of a particular structure can be useful starting vectors for the computation of the eigenvectors of a slightly altered structure. Similarly, the eigenvectors for a particular conformation of a protein can be useful starting vectors for the computation of the eigenvectors of a nearby conformation of the original protein [11, 23].

3. The algorithm works directly on the generalized eigenvalue problem and does not require conversion of (1.2) to the standard form of $\tilde{\boldsymbol{K}}\tilde{\boldsymbol{\phi}} = \omega^2\tilde{\boldsymbol{\phi}}$, for suitable choices of $\tilde{\boldsymbol{K}}$ and $\tilde{\boldsymbol{\phi}}$.

4. The algorithm does not require the computation of all eigenpairs.

5. The algorithm is easily parallelized in both shared and distributed memory processing schemes [4]. In fact, it was demonstrated in reference [4] that a parallel implementation of the basic subspace iteration method achieved solution times for (1.2) that increased linearly with respect to the number of eigenpairs sought, which is comparable to what can be observed using the Lanczos method.

When the subspace iteration method was first proposed in the 1970's, the finite element analyses of structures at the time typically incorporated only a few thousand degrees of freedom and required the computation of only a few dozen eigenpairs. However, in current analyses, structures with millions of degrees of freedom can be considered, and several hundred eigenpairs are frequently required. Since the Bathe subspace iteration method has been established as a reliable method for the solution of eigenvalue problems in computational mechanics, there is a significant interest in improving the original

methodology to make the procedure effective for the growing demands of today's analyses [16, 25]. Several schemes have been proposed to accelerate the convergence of the basic subspace iteration method, and we will briefly summarize some of the most relevant ones here to give the reader a general sense of what research has been done in this area.

In reference [3], the author proposes using over-relaxation procedures to modify how the iteration vectors are updated after the Rayleigh-Ritz analysis of the basic method. Let q denote the number of iteration vectors being used in the subspace iteration method. The over-relaxation strategy requires the computation of an over-relaxation factor (potentially one factor for each iteration vector) that depends on an accurate estimation of ω_{q+1}^2. Since ω_{q+1}^2 is not estimated directly in the subspace iteration method, it is necessary to approximate this value, which can be done indirectly by estimating the asymptotic convergence rates for some of the iteration vectors [6]. However, this approximation of ω_{q+1}^2 is not reliable until at least some of the iteration vectors have reached their asymptotic rate of convergence, which may not happen until after a few, or many, iterations. Therefore, this acceleration strategy cannot usually be applied during the initial iterations of the algorithm when acceleration is particularly valuable.

In reference [13], a two-phase method is proposed that combines the subspace iteration method with Rayleigh quotient iteration. The first phase of this strategy is to use a few iterations of the basic subspace iteration method to compute initial approximations of the target eigenvectors. If enough starting vectors are used in the subspace iteration phase, then it is possible to obtain good approximations of the target eigenvectors. These eigenvector approximations are subsequently used in the second phase as the inputs for Rayleigh quotient iteration. This hybrid proposal combines the ability of the subspace iteration method to provide quick approximations of the target eigenvectors with the cubic convergence rate of Rayleigh quotient iteration.

In references [6, 16, 28], a shifting technique of the form $\boldsymbol{K} - \alpha \boldsymbol{M}$ has been proposed using particular selections of the shift $\alpha \in \mathbb{R}$. When the subspace iteration method is applied to the shifted system $\boldsymbol{K} - \alpha \boldsymbol{M}$, the algorithm will converge most quickly to the eigenpairs with an eigenvalue near α. Shifting is a useful strategy because it makes it

possible to obtain faster convergence to higher frequency eigenpairs than would otherwise be possible operating only on the original, un-shifted, system. However, shifting requires computing the Cholesky decomposition of $\boldsymbol{K} - \alpha \boldsymbol{M}$, which incurs an additional computational cost. Therefore, an efficient shifting strategy needs to balance the cost of additional Cholesky decompositions with the benefit of accelerated convergence to the eigenpairs near the shift.

Of particular relevance to this work is the enriched subspace iteration method, which is described in references [17, 18]. The key observation of the enriched method is that the iteration vectors turn a certain amount towards the target subspace after the simultaneous inverse iteration step of the basic algorithm. Since Rayleigh-Ritz analysis is used to extract the best eigenvector approximations from the current subspace of iteration vectors, it is proposed that the current subspace can be improved by discarding some of the less effective iteration vectors in favor of turning vectors that point in the direction that the current subspace is turning towards the target subspace. The calculation of turning vectors is computationally inexpensive and their inclusion in the subspace results in significant computational savings compared to the basic subspace iteration method. Another appealing property of the enriched method is that the above over-relaxation, hybridization, and shifting acceleration strategies can be applied to the enriched method in the same manner that they are applied to the basic method.

In this thesis, we improve upon the enrichment procedure proposed in references [17, 18] for the acceleration of the basic subspace iteration method. Specifically, we develop the E^2 Bathe subspace iteration method, which uses turning vectors as well as our new turning-of-turning vectors to enrich the subspace of the current iteration by replacing less effective iteration vectors. Determining the turning-of-turning vectors is computationally inexpensive and often results in computational savings compared to the enriched method, particularly when relatively few iteration vectors are used in comparison to the number of eigenpairs that are sought. The E^2 method maintains the reliability of the basic subspace iteration method while also providing a superior theoretical asymptotic convergence rate compared to the basic and enriched subspace iteration methods.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

THE BATHE SUBSPACE ITERATION METHOD

In this chapter, we provide a review of the original Bathe subspace iteration method. We also discuss the enriched Bathe subspace iteration method that was recently proposed in references [18, 17]. The enriched method, which accelerates convergence of the basic method by using turning vectors to replace less effective iteration vectors, is particularly relevant to the E^2 algorithm that is developed in the subsequent chapter. The material in this chapter provides context for the novel material presented in Chapter 3.

The Bathe subspace iteration method is an iterative approach to solving the generalized eigenvalue problem. Specifically, given symmetric and positive definite matrices $\boldsymbol{K}$ and $\boldsymbol{M}$ of order n, the Bathe subspace iteration method can be used to compute the first p eigenpairs of the least dominant subspace of $\boldsymbol{K}$ and $\boldsymbol{M}$.

$$\begin{aligned} \boldsymbol{K}\boldsymbol{\Phi} = \boldsymbol{M}\boldsymbol{\Phi}\boldsymbol{\Omega}^2, \qquad \text{where } \boldsymbol{\Phi} &= \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_p \end{bmatrix} \\ \boldsymbol{\Omega}^2 &= \operatorname{diag}(\omega_1^2, \omega_2^2, \ldots, \omega_p^2) \end{aligned} \tag{2.1}$$

In (2.1), $\boldsymbol{\Phi} \in \mathbb{R}^{n \times p}$ and $\boldsymbol{\Omega}^2 \in \mathbb{R}^{p \times p}$. The generalized eigenvectors, $\boldsymbol{\Phi}$, of (2.1) are assumed to be normalized so that they are $\boldsymbol{M}$-orthonormal and $\boldsymbol{K}$-orthogonal, as indicated below:

$$\boldsymbol{\phi}_i^T \boldsymbol{M} \boldsymbol{\phi}_j = \delta_{ij} \quad \text{and} \quad \boldsymbol{\phi}_i^T \boldsymbol{K} \boldsymbol{\phi}_j = \omega_i^2 \delta_{ij}, \quad i, j = 1, 2, \ldots, p \tag{2.2}$$

In (2.2), δ_{ij} is the Kronecker delta function, which is defined as follows:

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{2.3}$$

Due to the assumption that $\boldsymbol{K}$ and $\boldsymbol{M}$ are both positive definite, all n generalized eigenvalues of $\boldsymbol{K}$ and $\boldsymbol{M}$ are positive and finite. Throughout this thesis, we will use the following convention for ordering the eigenvalues and the corresponding eigenvectors:

$$\begin{aligned} &0 < \omega_1^2 \leq \omega_2^2 \leq \cdots \leq \omega_p^2 \leq \cdots \leq \omega_n^2 < \infty \\ &\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_p, \ldots, \boldsymbol{\phi}_n \end{aligned} \tag{2.4}$$

The subspace iteration method is initialized with a set of $q > p$ linearly independent starting vectors $\boldsymbol{X}_0 \in \mathbb{R}^{n \times q}$, given by $\boldsymbol{X}_0 = \begin{bmatrix} \boldsymbol{x}_1^{(0)} & \boldsymbol{x}_2^{(0)} & \cdots & \boldsymbol{x}_{q-1}^{(0)} & \boldsymbol{x}_q^{(0)} \end{bmatrix}$. In general, the iteration vectors in the k^{th} iteration are denoted

$$\boldsymbol{X}_k = \begin{bmatrix} \boldsymbol{x}_1^{(k)} & \boldsymbol{x}_2^{(k)} & \cdots & \boldsymbol{x}_{q-1}^{(k)} & \boldsymbol{x}_q^{(k)} \end{bmatrix} \qquad \boldsymbol{X}_k \in \mathbb{R}^{n \times q} \tag{2.5}$$

Corresponding to each set of iteration vectors, $\boldsymbol{X}_k$, is a q-dimensional subspace defined by

$$E_k = \text{span}\left\{ \boldsymbol{x}_1^{(k)}, \boldsymbol{x}_2^{(k)}, \ldots, \boldsymbol{x}_{q-1}^{(k)}, \boldsymbol{x}_q^{(k)} \right\} \tag{2.6}$$

The target subspace of the q least dominant eigenvectors of (2.1) is denoted

$$E_\infty = \text{span}\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_q\} \tag{2.7}$$

The goal of the subspace iteration method is to produce a sequence of subspaces $E_0, E_1, E_2, \ldots, E_{\text{fin}}$ that converges to E_∞. If $E_0, E_1, E_2, \ldots, E_{\text{fin}}$ converge to E_∞, then E_{fin} can be used to obtain the required eigenvectors by the use of Ritz analysis, as demonstrated in reference [7]. The corresponding eigenvalue approximations are obtained from the eigenvector approximations by using the Rayleigh quotient. Since the Rayleigh quotient is used to obtain the eigenvalues, the error of the eigenvalue approximations is on

the order of the square of the error of the corresponding eigenvector approximations.

An important property of the subspace iteration method is that it is not necessary for the iteration vectors $\boldsymbol{x}_1^{(k)}, \boldsymbol{x}_2^{(k)}, \ldots, \boldsymbol{x}_{q-1}^{(k)}, \boldsymbol{x}_q^{(k)}$ to individually converge to an eigenvector of (2.1). Instead, it is only necessary to be able to obtain the target eigenvectors by linear combinations of the iteration vectors because the best eigenvector approximations will be extracted from the current subspace of iteration vectors during the Rayleigh-Ritz analysis. Consequently, if the starting iteration vectors happen to span the target subspace, then the Bathe subspace iteration method will converge in one iteration.

Moreover, even if the starting vectors do not span the target subspace, the number of iterations of the subspace iteration method that are required to reach convergence does not depend on the closeness of each iteration vector in $\boldsymbol{X}_0$ to a target eigenvector, but rather on the closeness of E_0 to E_∞. One of the reasons why the Bathe subspace iteration method is effective is based on the presumption that it is easier to find a starting subspace that is close to E_∞ than it is to find starting iteration vectors that are close to the required eigenvectors.

The steps of the Bathe subspace iteration method are as follows:

1. Select $q > p$ linearly independent starting vectors to create $\boldsymbol{X}_0 \in \mathbb{R}^{n \times q}$:

$$\boldsymbol{X}_0 = \begin{bmatrix} \boldsymbol{x}_1^{(0)} & \boldsymbol{x}_2^{(0)} & \cdots & \boldsymbol{x}_{q-1}^{(0)} & \boldsymbol{x}_q^{(0)} \end{bmatrix} \tag{2.8}$$

2. For $k = 1, 2, \ldots$, repeat Steps 2(a) – 2(c) below until convergence of E_k to E_∞:

   (a) Perform simultaneous inverse iteration to obtain $\overline{\boldsymbol{X}}_k \in \mathbb{R}^{n \times q}$ from $\boldsymbol{X}_{k-1}$:

$$\boldsymbol{K}\overline{\boldsymbol{X}}_k = \boldsymbol{M}\boldsymbol{X}_{k-1}, \quad k = 1, 2, \ldots \tag{2.9}$$

   (b) Perform the Rayleigh-Ritz procedure:

      i. Project the stiffness and mass matrices onto the current subspace $E_k =$

span $\{\overline{\boldsymbol{X}}_k\}$:

$$\begin{aligned} \boldsymbol{K}_k &= \overline{\boldsymbol{X}}_k^T \boldsymbol{K} \overline{\boldsymbol{X}}_k, \qquad & \boldsymbol{K}_k \in \mathbb{R}^{q \times q} \\ \boldsymbol{M}_k &= \overline{\boldsymbol{X}}_k^T \boldsymbol{M} \overline{\boldsymbol{X}}_k, \qquad & \boldsymbol{M}_k \in \mathbb{R}^{q \times q} \end{aligned} \tag{2.10}$$

ii. Determine $\boldsymbol{Q}_k \in \mathbb{R}^{q \times q}$ and $\boldsymbol{\Omega}_k^2 \in \mathbb{R}^{q \times q}$ by solving the generalized eigenvalue problem of the projected operators:

$$\boldsymbol{K}_k \boldsymbol{Q}_k = \boldsymbol{M}_k \boldsymbol{Q}_k \boldsymbol{\Omega}_k^2 \tag{2.11}$$

iii. Compute an improved approximation of the eigenvectors using the Ritz coordinates obtained in (2.11):

$$\boldsymbol{X}_k = \overline{\boldsymbol{X}}_k \boldsymbol{Q}_k \tag{2.12}$$

(c) Check for convergence of the eigenvalues using the criterion in (2.13), which was given in reference [5].

$$\left[1 - \frac{\left(\omega_i^{2,(k)}\right)^2}{\left(\boldsymbol{q}_i^{(k)}\right)^T \boldsymbol{q}_i^{(k)}} \right]^{1/2} \leq tol \tag{2.13}$$

In (2.13), $\omega_i^{2,(k)}$ denotes the approximation of ω_i^2 in the k^{th} iteration and $\boldsymbol{q}_i^{(k)}$ is the i^{th} column of $\boldsymbol{Q}_k$.

3. After convergence to E_∞, perform the Sturm sequence check to ensure that all required eigenpairs have been computed. A thorough discussion of how to apply the Sturm sequence check is provided in references [9, 5].

In the algorithm above, it is assumed that the iteration vectors are sorted so that the iteration vector converging to the i^{th} eigenvector is stored in the i^{th} column of $\boldsymbol{X}_k$. Additionally, note that (2.11) requires the computation of the complete eigensystem of the projected operators $\boldsymbol{K}_k$ and $\boldsymbol{M}_k$. Using the Jacobi iteration method to solve the order q eigenvalue problem, the computational cost of this step is $\mathcal{O}(q^3)$, which is

relatively inexpensive compared to the cost of other steps in the algorithm since $q \ll n$.

We recommend that the user preset the convergence tolerance used in (2.13) following the guideline $tol = 10^{-2t}$, where $t \in \mathbb{Z}$ is the number of digits to which the eigenvectors should be accurate, and $2t$ is the number of digits to which the eigenvalues should be accurate [5]. The eigenvalue approximations are accurate to twice as many digits as the corresponding eigenvector approximations because the eigenvalue approximations are computed using the Rayleigh quotient, as discussed previously.

If the starting vectors are not deficient in any of the first q generalized eigenvectors, then the following convergence limits are observed, where $\boldsymbol{\Phi}$ and $\boldsymbol{\Omega}^2$ denote the first q eigenvectors and eigenvalues, respectively, of the least dominant subspace of $\boldsymbol{K}$ and $\boldsymbol{M}$:

$$\lim_{k\to\infty} \boldsymbol{X}_k = \boldsymbol{\Phi} \quad \text{and} \quad \lim_{k\to\infty} \boldsymbol{\Omega}_k^2 = \boldsymbol{\Omega}^2 \tag{2.14}$$

The purpose of the Sturm sequence check is to verify that the algorithm has, indeed, converged to all of the target eigenpairs. An eigenpair can be missed if the starting vectors are all deficient in that particular eigenvector. If the Sturm sequence check reveals that there are missing eigenpairs, we recommend appending random vectors to the current set of iteration vectors. Then, continue the subspace iteration method with the higher-dimensional subspace. Note that it is only necessary to continue the subspace iteration on the non-converged vectors. Therefore, converging to the missing eigenpairs can be quick if only a few of the target eigenpairs are found to be missing after performing the Sturm sequence check.

However, we emphasize that while it is theoretically possible for the subspace iteration method to not converge to one or more of the target eigenpairs, if q is sufficiently larger than p, then the method rarely fails to converge to any of the target eigenpairs in practice. In reference [8], it is recommended to determine the number of iteration vectors, q, by

$$q = \max\{2p, p+8\} \tag{2.15}$$

If q is selected by (2.15), then it is very unlikely that the starting vectors will all be

$\boldsymbol{M}$-orthogonal to any of the target eigenvectors, especially if the starting vectors are selected using the procedure described in reference [5] or if random vectors are used as the starting vectors.

An interesting observation regarding the history of the subspace iteration method is that, originally, it was recommended to select q according to $q = \min\{2p, p+8\}$, as stated in reference [7]. Selecting q in this manner was appropriate for the analyses that occurred early in the history of the method when comparatively small problems were considered, only a few dozen eigenpairs were sought, and computers were much less powerful [4]. However, in reference [4], the author uses the fact that the eigenvalues of structures frequently increase according to some known or guessed functional form, which can be used to determine an optimal value of q. The resulting analysis given in reference [4] ultimately concludes that the recommendation for selecting q that is given in (2.15) is satisfactory for general use.

It was shown in [3] that if E_k is sufficiently close to E_∞, then the following convergence rates are observed for the i^{th} iteration vector to $\boldsymbol{\phi}_i$ and for the i^{th} eigenvalue to ω_i^2:

$$\begin{aligned} &\boldsymbol{x}_i^{(k)} \to \boldsymbol{\phi}_i \text{ at a rate on the order of } \frac{\omega_i^2}{\omega_{q+1}^2} \\ &\omega_i^{2,(k)} \to \omega_i^2 \text{ at a rate on the order of } \left(\frac{\omega_i^2}{\omega_{q+1}^2}\right)^2 \end{aligned} \tag{2.16}$$

Since the theoretical convergence rates of (2.16) require E_k to be sufficiently close to E_∞, they are known as asymptotic convergence rates, and we note that the asymptotic rates might not be observed during the initial iterations of the algorithm. Hence, there is also significant interest in beginning the subspace iteration with high-quality starting vectors that reach their asymptotic convergence rates immediately or after only a few iterations.

The asymptotic convergence rate of the i^{th} eigenpair, for i in the interval $1 \le i \le q$, is fastest when the ratio $\omega_i^2 \big/ \omega_{q+1}^2$ is small. Consequently, the smallest eigenvalues have the fastest asymptotic convergence rates. Therefore, because of the ordering in (2.4), ω_1^2 converges most quickly, and ω_p^2 is the slowest of the target eigenvalues to converge, with

an asymptotic convergence rate of $\left(\omega_p^2 \Big/ \omega_{q+1}^2\right)^2$.

However, all eigenpairs will have faster asymptotic convergence rates as $q \to n$ unless $\omega_{q+1}^2 = \omega_{q+2}^2 = \cdots = \omega_n^2$. In this unlikely scenario, of course, the ratio $\omega_i^2 \Big/ \omega_{q+1}^2$ does not change as q approaches n. But, assuming that the eigenvalues are reasonably well-spaced, then there is an interesting trade-off that is a consequence of the convergence rate in (2.16):

1. If a large value for q is selected, then each of the target eigenpairs has a faster asymptotic convergence rate. However, each iteration of the algorithm is more computationally expensive due to using more iteration vectors. The consequence of using the additional iteration vectors is that each iteration requires more central processing unit (CPU) time to complete.

2. Conversely, if a small value for q is selected, then each of the target eigenpairs has a slower asymptotic convergence rate, but each iteration of the algorithm is less computationally expensive due to using fewer iteration vectors and therefore each iteration costs less CPU time.

The optimal value of q with respect to minimizing the total CPU time required to reach convergence is specific to each problem and can depend on the quality of the starting vectors and how well-spaced the target eigenvalues are, among other factors. The importance of this trade-off is examined in the test problems of Chapter 3.

We conclude this chapter with a discussion of the enriched subspace iteration method, which was first proposed in references [18, 17]. As indicated in Figure 2-1, the enriched subspace method is closely related to the basic method. In fact, the only difference between the two methods is in how the iteration vectors undergo the inverse iteration step, and consequently, how the vectors that define the next subspace are determined.

In the basic method, the inverse iteration step is $\overline{\boldsymbol{X}}_k = \boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{X}_{k-1}$. However, in the enriched method, the iteration vectors in $\boldsymbol{X}_{k-1}$ are first partitioned as follows:

$$\boldsymbol{X}_{k-1} = \begin{bmatrix} \boldsymbol{\Phi}_{k-1} & \boldsymbol{X}_{k-1}^a & \boldsymbol{X}_{k-1}^b \end{bmatrix} \qquad \begin{array}{l} \boldsymbol{\Phi}_{k-1} \in \mathbb{R}^{n \times p_{k-1}},\ \boldsymbol{X}_{k-1}^a \in \mathbb{R}^{n \times r_k^a}, \\ \boldsymbol{X}_{k-1}^b \in \mathbb{R}^{n \times r_k^b} \end{array} \tag{2.17}$$

In (2.17), p_{k-1} is the number of converged vectors after $k-1$ iterations, with $p_0 = 0$. The remaining iteration vectors are split between $\boldsymbol{X}_{k-1}^a$ and $\boldsymbol{X}_{k-1}^b$. The partition sizes satisfy $q = p_{k-1} + r_k^a + r_k^b$. The iteration vectors in $\boldsymbol{\Phi}_{k-1}$ no longer undergo the inverse iteration step since they have already converged within the pre-specified tolerance, which is important for reducing the computational expense of the algorithm.

Then, in the enriched method, the inverse iteration step is, at first, only applied to $\boldsymbol{X}_{k-1}^a$, as opposed to the entire $\boldsymbol{X}_{k-1}$ as is the case in the basic method. Thus,

$$\overline{\boldsymbol{X}}_k^a = \boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{X}_{k-1}^a, \qquad \overline{\boldsymbol{X}}_k^a \in \mathbb{R}^{n \times r_k^a} \tag{2.18}$$

The key insight of the enriched method is that in going from $\boldsymbol{X}_{k-1}^a$ to $\overline{\boldsymbol{X}}_k^a$, the iteration vectors have turned a certain amount towards the target subspace. Turning vectors are identified from the iteration vectors that turned significantly from $\boldsymbol{X}_{k-1}^a$ to $\overline{\boldsymbol{X}}_k^a$. Then, these turning vectors are used to replace less effective iteration vectors in $\boldsymbol{X}_{k-1}^b$ that are not converging to one of the target eigenvectors.

The updated $\boldsymbol{X}_{k-1}^b$, which now includes the turning vectors, is then $\boldsymbol{M}$-orthogonalized and normalized, and the result is called $\boldsymbol{Y}_{k-1} \in \mathbb{R}^{n \times r_k^b}$. Next, the newly formed $\boldsymbol{Y}_{k-1}$ undergoes inverse iteration resulting in:

$$\overline{\boldsymbol{Y}}_k = \boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{Y}_{k-1}, \qquad \overline{\boldsymbol{Y}}_k \in \mathbb{R}^{n \times r_k^b} \tag{2.19}$$

After (2.19), the vectors that were previously known as turning vectors are now known as forward turning vectors. The forward turning vectors point in the direction that the current subspace is turning towards the target subspace. Finally, the new q-dimensional subspace is defined by

$$E_k = \text{span}\left\{\boldsymbol{\Phi}_{k-1},\ \overline{\boldsymbol{X}}_k^a,\ \overline{\boldsymbol{Y}}_k\right\} \tag{2.20}$$

As indicated in Figure 2-1, after the new subspace has been defined, the rest of the enriched algorithm proceeds according to the steps of the basic method. The enrichment procedure is a simple addition to the basic subspace iteration method that is very effective in improving the asymptotic convergence rate of the basic method.

The forward turning vectors are noteworthy because they underwent inverse iteration twice: once in (2.18) and a second time in (2.19). However, it is important to recognize that even though the forward turning vectors underwent inverse iteration twice, the total number of vectors undergoing inverse iteration within a single iteration of the algorithm is still $q - p_{k-1}$ because an iteration vector was discarded for each turning vector that was selected. Hence, no more vectors undergo inverse iteration in the enriched method than in the basic method. Consequently, the only additional cost of the enriched method is the determination of the turning vectors.

Lastly, in reference [18], the author performed a simplified convergence analysis and obtained the following asymptotic convergence rates for the enriched subspace iteration method for the case when $q = 2p$ and all possible turning vectors are used in each iteration:

$$\begin{aligned} &\boldsymbol{x}_i^{(k)} \to \boldsymbol{\phi}_i \text{ at a rate on the order of } \left(\frac{\omega_i^2}{\omega_{q+1}^2}\right)^2 \qquad 1 \le i \le p \\ &\omega_i^{2,(k)} \to \omega_i^2 \text{ at a rate on the order of } \left(\frac{\omega_i^2}{\omega_{q+1}^2}\right)^4 \qquad 1 \le i \le p \end{aligned} \tag{2.21}$$

The computational savings of the enriched subspace iteration method, as reported in reference [18] make the enrichment procedure an important acceleration strategy for the basic subspace iteration method.

In this thesis, we have extended the enrichment procedure to now include two enrichments of the subspace during each iteration of the algorithm. The first enrichment entails computing turning vectors by the process just described. The second enrichment entails computing turning-of-turning vectors using our new process, which will be described in Chapter 3. The result is a further improvement to the asymptotic convergence rates of the subspace iteration method beyond the convergence rates reported in (2.21).

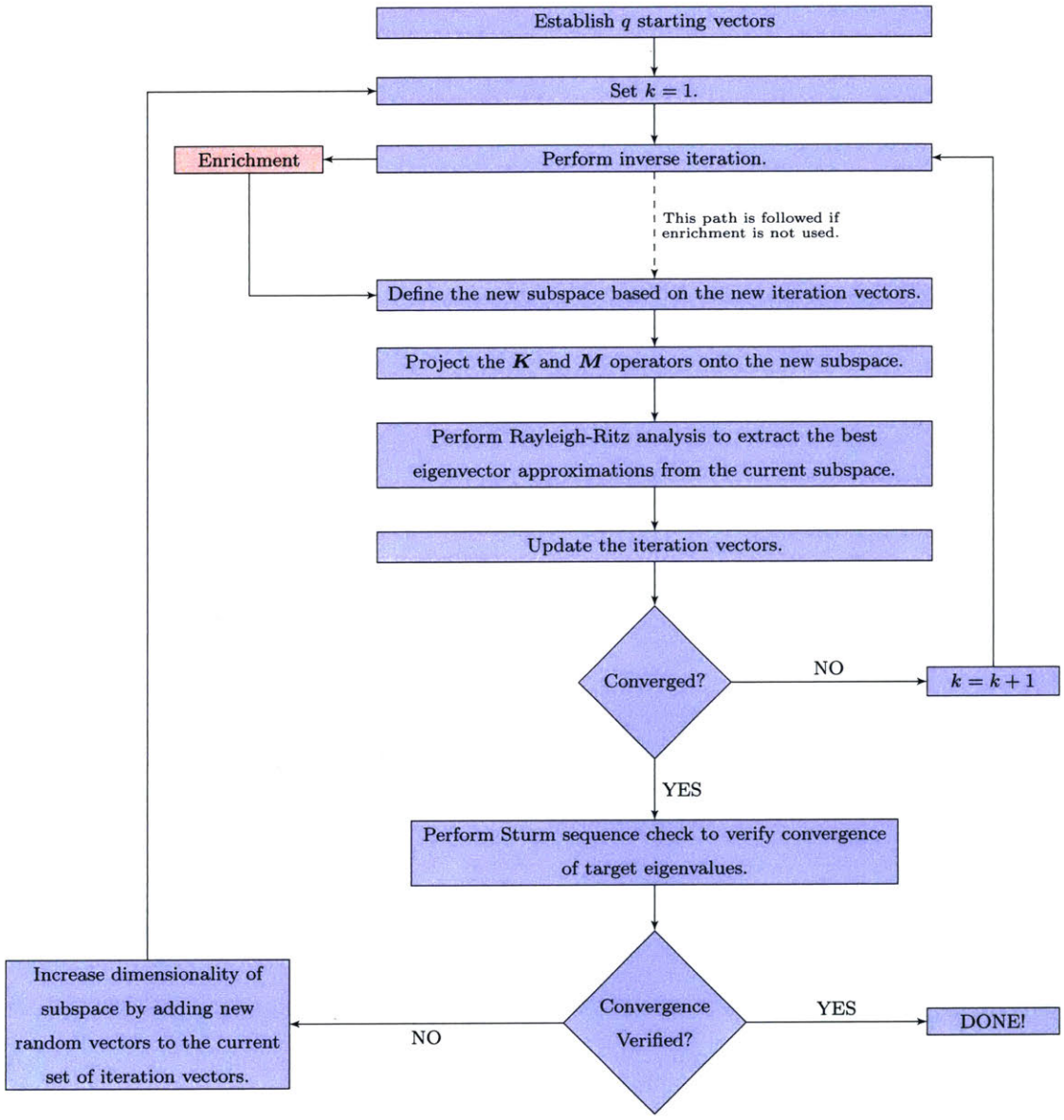

Figure 2-1: Flow chart depicting the relationship between the basic and enriched subspace iteration methods. The blue boxes are the steps of the basic method. The red box is the enrichment step in which turning vectors are calculated and used to replace some of the less effective iteration vectors. In the basic method, there is no enrichment step. Instead, the algorithm immediately defines the new subspace after the simultaneous inverse iteration step. Note: while it is theoretically possible to miss convergence to one of the target eigenvectors, for example if all of the starting vectors are $\boldsymbol{M}$-orthogonal to one of the target eigenvectors, this situation rarely occurs in practice if a large enough q is selected. Using Equation (2.15) to determine q is usually sufficient in practice.

Chapter 3

THE E^2 BATHE SUBSPACE ITERATION METHOD

A key observation of the subspace iteration method is that after each iteration, the iteration vectors have turned a certain amount towards the target subspace. This turning occurs primarily as a result of the inverse iteration step. Another important observation of the basic method is that due to the use of Ritz analysis, which extracts the best eigenvector approximations from the current subspace, faster convergence can be obtained if a higher quality subspace is available.

The insight of the work done in references [17, 18] is that a better subspace can be obtained in each iteration of the basic method by replacing some of the less effective iteration vectors with turning vectors. The turning vectors, and their corresponding forward turning vectors, enrich the subspace by incorporating information about the direction that the current subspace is turning towards the target subspace. This additional turning information is useful since Ritz analysis extracts the best eigenvector approximations from the possible linear combinations of the iteration vectors. The turning vectors are simple to calculate, yet they have a significant speed-up effect when they are used to enrich the subspace of the basic subspace iteration method.

In this chapter, we present a novel extension of the enriched subspace iteration method, which we refer to as the E^2 Bathe subspace iteration method. The idea of the

E^2 method is to measure the turning of the turning vectors after they undergo inverse iteration as part of the enriched method. Then, the turning vectors that are determined to have turned significantly after the inverse iteration step are used to compute our new turning-of-turning vectors.

Enriching the subspace of the basic subspace iteration method by including turning vectors as well as our new turning-of-turning vectors is the central idea for obtaining the improved asymptotic convergence rate that the E^2 method provides.

3.1 The New Algorithm

Like the basic and enriched subspace iteration methods, the E^2 method is used to solve the generalized, symmetric-definite eigenvalue problem of (3.1).

$$\boldsymbol{K}\boldsymbol{\phi} = \omega^2 \boldsymbol{M}\boldsymbol{\phi} \tag{3.1}$$

The finite element matrices $\boldsymbol{K}$ and $\boldsymbol{M}$ are symmetric, positive-definite, and of order n. We are particularly interested in eigenvalue problems when n is large, $\boldsymbol{K}$ and $\boldsymbol{M}$ are sparse, and only the first $p \ll n$ least dominant eigenpairs are sought. The desired eigenpairs are written compactly below.

$$\begin{aligned} \boldsymbol{K}\boldsymbol{\Phi} = \boldsymbol{M}\boldsymbol{\Phi}\boldsymbol{\Omega}^2, \qquad \text{where } \boldsymbol{\Phi} &= \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_p \end{bmatrix} \\ \boldsymbol{\Omega}^2 &= \operatorname{diag}(\omega_1^2, \omega_2^2, \ldots, \omega_p^2) \end{aligned} \tag{3.2}$$

As before, the ordering of the eigenpairs is assumed to be

$$\begin{aligned} &0 < \omega_1^2 \leq \omega_2^2 \leq \cdots \leq \omega_p^2 \leq \cdots \leq \omega_n^2 < \infty \\ &\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_p, \ldots, \boldsymbol{\phi}_n \end{aligned} \tag{3.3}$$

3.1.1 Steps of the E^2 Bathe Subspace Iteration Method

The E^2 algorithm is initialized with a set of linearly independent starting vectors, denoted $\boldsymbol{X}_0 \in \mathbb{R}^{n \times q}$, where $q > p$ is the number of iteration vectors that will be used to obtain

the convergence of the p least-dominant eigenpairs.

$$\boldsymbol{X}_0 = \left[\boldsymbol{x}_1^{(0)} \quad \boldsymbol{x}_2^{(0)} \quad \cdots \quad \boldsymbol{x}_q^{(0)}\right] \tag{3.4}$$

One commonly-used procedure to generate the starting vectors that is based on exciting the target degrees of freedom is described in reference [5]. Alternatively, initializing the starting vectors with q random vectors can be a useful strategy for reducing the risk of the starting vectors being $\boldsymbol{M}$-orthogonal to any one of the target eigenvectors. If all of the starting vectors are $\boldsymbol{M}$-orthogonal to a target eigenvector, then the E^2 method will not converge to that eigenvector. Situations such as this are why the Sturm sequence check is used to ensure that no eigenpairs have been missed after the algorithm has reported convergence. However, we note that this situation is rarely encountered in practice if q is sufficiently larger than p. The recommendation of reference [8] is to select q according to

$$q = \max\{2p, p+8\} \tag{3.5}$$

Selecting q according to (3.5) makes it very unlikely that the starting vectors will all be $\boldsymbol{M}$-orthogonal to any of the target eigenvectors, especially if the starting vectors are selected using the procedure described in reference [5] or if all of the starting vectors are chosen as random vectors.

The starting q-dimensional subspace is $E_0 = \text{span}\left\{\boldsymbol{x}_1^{(0)}, \boldsymbol{x}_2^{(0)}, \ldots, \boldsymbol{x}_q^{(0)}\right\} = \text{span}\{\boldsymbol{X}_0\}$. Selecting good starting vectors is important because the number of iterations that will be required to reach convergence depends on the closeness of E_0 to the target subspace $E_\infty = \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_q\}$. It is noteworthy that if E_0 and E_∞ span the same space, then the subspace iteration method will converge in one iteration, even if none of the vectors in $\boldsymbol{X}_0$ are any of the target eigenvectors.

Additionally, the selection of q affects the number of iterations that will be required to reach convergence and also the CPU time required until convergence. The larger q is, the fewer number of iterations will be required to reach convergence. However, as q increases, so does the computational burden of each iteration, which results in each

iteration requiring more CPU time to complete. For general use, the recommendation of (3.5) seems to be practical. However, in Section 3.4, we examine the convergence results of the enriched and E^2 methods using different values of q.

Prior to beginning the main loop of the E^2 subspace iteration algorithm, the stiffness matrix is factorized into $\boldsymbol{K} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$. This decomposition of $\boldsymbol{K}$ is important since the algorithm requires solving linear systems with the coefficient matrix $\boldsymbol{K}$. Additionally, before beginning the algorithm, the starting vectors are $\boldsymbol{M}$-orthonormalized by performing an iteration of the basic subspace iteration method. After the $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ decomposition of $\boldsymbol{K}$ and the $\boldsymbol{M}$-orthonormalization of the starting vectors, the following steps are repeated until convergence of p eigenpairs.

For $k = 1, 2, \ldots,$ repeat the following steps until convergence:

1. Partition the iteration vectors as follows:

$$\boldsymbol{X}_{k-1} = \begin{bmatrix} \boldsymbol{\Phi}_{k-1} & \boldsymbol{X}^a_{k-1} & \boldsymbol{X}^b_{k-1} & \boldsymbol{X}^c_{k-1} \end{bmatrix}, \qquad \boldsymbol{X}_{k-1} \in \mathbb{R}^{n \times q} \tag{3.6}$$

   Notes on step 1:

   - The vectors stored in $\boldsymbol{\Phi}_{k-1} \in \mathbb{R}^{n \times p_{k-1}}$ are the converged iteration vectors. The number of converged iteration vectors at the beginning of the k^{th} iteration is denoted p_{k-1}, and the algorithm is initialized with $p_0 = 0$. $\boldsymbol{X}^a_{k-1}$ contains the iteration vectors that will be the first set of vectors to undergo simultaneous inverse iteration and will then be used to determine the turning vectors. The turning vectors will be used to replace less effective iteration vectors in $\boldsymbol{X}^b_{k-1}$. Turning-of-turning vectors will replace less effective iteration vectors in $\boldsymbol{X}^c_{k-1}$.
   - The number of iteration vectors stored in $\boldsymbol{X}^a_{k-1}$, $\boldsymbol{X}^b_{k-1}$, and $\boldsymbol{X}^c_{k-1}$ are r^a_k, r^b_k and r^c_k, respectively, and can be determined by the user subject to the

condition in (3.7).

$$q = p_{k-1} + r_k^a + r_k^b + r_k^c \tag{3.7}$$

2. Perform simultaneous inverse iteration on $\boldsymbol{X}_{k-1}^a$ to obtain $\overline{\boldsymbol{X}}_k^a$:

$$\overline{\boldsymbol{X}}_k^a = \boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{X}_{k-1}^a, \qquad \overline{\boldsymbol{X}}_k^a \in \mathbb{R}^{n \times r_k^a} \tag{3.8}$$

3. Let $t_{k,1} \in \mathbb{Z}$ denote the number of turning vectors in the k^{th} iteration. Initially, $t_{k,1} = 0$. Identify the turning vectors by repeating steps 3(a) and 3(b) for $i = r_k^a, r_k^a - 1, \ldots, \max\left\{1, r_k^a - r_k^b + 1\right\}$.

   (a) Let $\overline{\boldsymbol{x}}_i^{a,(k)}$ denote the i^{th} column of $\overline{\boldsymbol{X}}_k^a$, and let $\alpha_i \in \mathbb{R}$ denote the amount of turning of $\overline{\boldsymbol{x}}_i^{a,(k)}$ from $\boldsymbol{x}_i^{a,(k-1)}$, which is calculated by:

$$\hat{\boldsymbol{x}}_i = \overline{\boldsymbol{x}}_i^{a,(k)} - \underbrace{\boldsymbol{X}_{k-1}\left(\boldsymbol{X}_{k-1}^T\boldsymbol{M}\overline{\boldsymbol{x}}_i^{a,(k)}\right)}_{\substack{\text{The components of } \overline{\boldsymbol{x}}_i^{a,(k)} \\ \text{in the direction of the} \\ \text{previous subspace}}} - \underbrace{\sum_{j=1}^{t_{k,1}} \boldsymbol{u}_j\left(\boldsymbol{u}_j^T\boldsymbol{M}\overline{\boldsymbol{x}}_i^{a,(k)}\right)}_{\substack{\text{The components of } \overline{\boldsymbol{x}}_i^{a,(k)} \\ \text{in the direction of the} \\ \text{already-selected} \\ \text{turning vectors}}} \tag{3.9}$$

$$\alpha_i = \frac{\hat{\boldsymbol{x}}_i^T\boldsymbol{M}\hat{\boldsymbol{x}}_i}{\left(\overline{\boldsymbol{x}}_i^{a,(k)}\right)^T\boldsymbol{M}\overline{\boldsymbol{x}}_i^{a,(k)}}$$

   (b) Let $tolt > 0$ denote the tolerance used to determine if $\overline{\boldsymbol{x}}_i^{a,(k)}$ has turned enough to qualify as a turning vector. If $\alpha_i \leq tolt$, then $\overline{\boldsymbol{x}}_i^{a,(k)}$ is not a turning vector and proceed to the next i. Otherwise, $\overline{\boldsymbol{x}}_i^{a,(k)}$ is a turning vector and do the following:

$$\begin{aligned} t_{k,1} &= t_{k,1} + 1 \\ \boldsymbol{u}_{t_{k,1}} &= \frac{\hat{\boldsymbol{x}}_i}{\sqrt{\hat{\boldsymbol{x}}_i^T\boldsymbol{M}\hat{\boldsymbol{x}}_i}} \\ \boldsymbol{v}_{t_{k,1}} &= \overline{\boldsymbol{x}}_i^{a,(k)} \end{aligned} \tag{3.10}$$

   Proceed to the next i.

Notes on step 3:

- The iteration vectors are considered in reverse order in this step because of how the iteration vectors are assumed to be ordered. Recall the iteration vectors are ordered as follows: the iteration vector converging to $\boldsymbol{\phi}_1$ is stored in the first column of $\boldsymbol{X}_{k-1}$, the iteration vector converging to $\boldsymbol{\phi}_2$ is stored in the second column of $\boldsymbol{X}_{k-1}$, etc. Therefore, the first few vectors in $\overline{\boldsymbol{X}}_k^a$ and $\boldsymbol{X}_{k-1}^a$ are presumably close to convergence and do not turn very much from $\boldsymbol{X}_{k-1}^a$ to $\overline{\boldsymbol{X}}_k^a$.

4. Let $t_{k,1}$ be the final value of $t_{k,1}$ obtained in step 3. Let $\boldsymbol{x}_i^{b,(k-1)}$ denote the i^{th} column of $\boldsymbol{X}_{k-1}^b$. Then, construct $\boldsymbol{Y}_{k-1} \in \mathbb{R}^{n \times r_k^b}$ by

$$\boldsymbol{Y}_{k-1} = \left[\boldsymbol{x}_1^{b,(k-1)} \quad \cdots \quad \boldsymbol{x}_{r_k^b - t_{k,1}}^{b,(k-1)} \;\middle|\; \boldsymbol{v}_1 \quad \cdots \quad \boldsymbol{v}_{t_{k,1}} \right] \tag{3.11}$$

Notes on step 4:

- The first $r_k^b - t_{k,1}$ columns of $\boldsymbol{Y}_{k-1}$ are copied from the first $r_k^b - t_{k,1}$ iteration vectors of $\boldsymbol{X}_{k-1}^b$. The last $t_{k,1}$ vectors in $\boldsymbol{Y}_{k-1}$ are vectors from $\overline{\boldsymbol{X}}_k^a$ and will become the turning vectors.

5. $\boldsymbol{M}$-orthogonalize $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t_{k,1}}$. For $i = 1, \ldots, t_{k,1}$, do the following in (3.12) and (3.13):

$$\begin{aligned} \tilde{\boldsymbol{x}}_i = \boldsymbol{v}_i &- \underbrace{\boldsymbol{X}_{k-1}^a \left((\boldsymbol{X}_{k-1}^a)^T \boldsymbol{M} \boldsymbol{v}_i \right)}_{\substack{\boldsymbol{M}\text{-orthogonalize to} \\ \text{the vectors in } \boldsymbol{X}_{k-1}^a}} - \underbrace{\boldsymbol{\Phi}_{k-1} \left(\boldsymbol{\Phi}_{k-1}^T \boldsymbol{M} \boldsymbol{v}_i \right)}_{\substack{\boldsymbol{M}\text{-orthogonalize to} \\ \text{the vectors in } \boldsymbol{\Phi}_{k-1}}} \\ &- \underbrace{\sum_{j=1}^{r_k^b - t_{k,1} + i - 1} \boldsymbol{y}_j^{(k-1)} \left(\left(\boldsymbol{y}_j^{(k-1)} \right)^T \boldsymbol{M} \boldsymbol{v}_i \right)}_{\substack{\boldsymbol{M}\text{-orthogonalize to the first} \\ r_k^b - t_{k,1} \text{ vectors in } \boldsymbol{X}_{k-1}^b \text{ and} \\ \text{to the first } i-1 \text{ turning vectors}}} - \underbrace{\boldsymbol{X}_{k-1}^c \left((\boldsymbol{X}_{k-1}^c)^T \boldsymbol{M} \boldsymbol{v}_i \right)}_{\substack{\boldsymbol{M}\text{-orthogonalize to} \\ \text{the vectors in } \boldsymbol{X}_{k-1}^c}} \end{aligned} \tag{3.12}$$

$$\boldsymbol{y}_{r_k^b - t_{k,1} + i}^{(k-1)} = \frac{\tilde{\boldsymbol{x}}_i}{\sqrt{\tilde{\boldsymbol{x}}_i^T \boldsymbol{M} \tilde{\boldsymbol{x}}_i}} \tag{3.13}$$

Notes on step 5:

- The vectors $\tilde{\boldsymbol{x}}_i$, and their normalizations in (3.13), are the desired turning vectors.
- The purpose of the $\boldsymbol{M}$-orthonormalization in this step is to obtain a more numerically stable basis for $\boldsymbol{Y}_{k-1}$, which is necessary since $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t_{k,1}}$ can be very parallel after the inverse iteration of step 2.

6. Perform simultaneous inverse iteration on $\boldsymbol{Y}_{k-1}$ to obtain $\overline{\boldsymbol{Y}}_k$:

$$\overline{\boldsymbol{Y}}_k = \boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{Y}_{k-1}, \qquad \overline{\boldsymbol{Y}}_k \in \mathbb{R}^{n \times r_k^b} \tag{3.14}$$

Notes on step 6:

- The first $r_k^b - t_{k,1}$ vectors of $\overline{\boldsymbol{Y}}_k$ have now undergone inverse iteration for the first time in the k^{th} iteration, which they would have done in the basic method, as well. However, the last $t_{k,1}$ vectors of $\overline{\boldsymbol{Y}}_k$ have now undergone inverse iteration twice in this iteration, which is the distinction between the enriched algorithm and the basic method.
- The last $t_{k,1}$ vectors of $\overline{\boldsymbol{Y}}_k$ are known as the forward turning vectors. The forward turning vectors are the key ingredient in the enriched method for accelerating the convergence of the basic method.

7. Let $t_{k,2} \in \mathbb{Z}$ denote the number of vectors that turned significantly from $\boldsymbol{Y}_{k-1}$ to $\overline{\boldsymbol{Y}}_k$. Initially, $t_{k,2} = 0$. In this step, we will identify additional turning vectors (similar to the turning vectors identified in step 3) and we will also identify our new turning-of-turning vectors.

   The new turning vectors and turning-of-turning vectors are identified by repeating steps 7(a) and 7(b) for $i = r_k^b, r_k^b - 1, \ldots, \max\left\{1, r_k^b - r_k^c + 1\right\}$.

   (a) Let $\overline{\boldsymbol{y}}_i^{(k)}$ denote the i^{th} column of $\overline{\boldsymbol{Y}}_k$, and let $\alpha_i \in \mathbb{R}$ denote the amount of

turning of $\overline{\boldsymbol{y}}_i^{(k)}$, which is calculated by:

$$\begin{aligned}\hat{\boldsymbol{y}}_i = \overline{\boldsymbol{y}}_i^{(k)} &- \underbrace{\begin{bmatrix}\boldsymbol{\Phi}_{k-1} & \boldsymbol{X}_{k-1}^a & \boldsymbol{Y}_{k-1} & \boldsymbol{X}_{k-1}^c\end{bmatrix}\begin{bmatrix}\boldsymbol{\Phi}_{k-1}^T \\ (\boldsymbol{X}_{k-1}^a)^T \\ \boldsymbol{Y}_{k-1}^T \\ (\boldsymbol{X}_{k-1}^c)^T\end{bmatrix}\boldsymbol{M}\overline{\boldsymbol{y}}_i^{(k)}}_{\substack{\text{The components of } \overline{\boldsymbol{y}}_i^{(k)} \\ \text{in the direction of the} \\ \text{previous subspace}}} \\ &- \underbrace{\sum_{j=1}^{t_{k,2}} \boldsymbol{u}_j \left(\boldsymbol{u}_j^T \boldsymbol{M}\overline{\boldsymbol{y}}_i^{(k)}\right)}_{\substack{\text{The components of } \overline{\boldsymbol{y}}_i^{(k)} \\ \text{in the direction of the} \\ \text{already-selected turning} \\ \text{and turning vectors of this step}}} \\ \alpha_i &= \frac{\hat{\boldsymbol{y}}_i^T \boldsymbol{M}\hat{\boldsymbol{y}}_i}{\left(\overline{\boldsymbol{y}}_i^{(k)}\right)^T \boldsymbol{M}\overline{\boldsymbol{y}}_i^{(k)}}\end{aligned} \tag{3.15}$$

(b) Let $tolt' > 0$ denote the tolerance used to determine if $\overline{\boldsymbol{y}}_i^{(k)}$ has turned enough to qualify as a turning (or turning-of-turning) vector. If $\alpha_i \leq tolt'$, then $\overline{\boldsymbol{y}}_i^{(k)}$ is not a turning (or turning-of-turning) vector and proceed to the next i. Otherwise, $\overline{\boldsymbol{y}}_i^{(k)}$ is a turning (or turning-of-turning) vector and do the following:

$$\begin{aligned} t_{k,2} &= t_{k,2} + 1 \\ \boldsymbol{u}_{t_{k,2}} &= \frac{\hat{\boldsymbol{y}}_i}{\sqrt{\hat{\boldsymbol{y}}_i^T \boldsymbol{M}\hat{\boldsymbol{y}}_i}} \\ \boldsymbol{w}_{t_{k,2}} &= \overline{\boldsymbol{y}}_i^{(k)}. \end{aligned} \tag{3.16}$$

Proceed to the next i.

Notes on step 7:

- The distinction between turning vectors and turning-of-turning vectors is as follows. A vector in $\overline{\boldsymbol{Y}}_k$ is a turning vector if it satisfies the following two requirements:

- The vector must have began the k^{th} iteration in $\boldsymbol{X}_{k-1}^{b}$ and must have been copied into $\boldsymbol{Y}_{k-1}$ during step 4.
- The vector must have turned significantly from $\boldsymbol{Y}_{k-1}$ to $\overline{\boldsymbol{Y}}_k$.

A vector in $\overline{\boldsymbol{Y}}_k$ is a turning-of-turning vector if it satisfies the following three requirements:

- The vector must have began the k^{th} iteration in $\boldsymbol{X}_{k-1}^{a}$.
- The vector must have been selected as a turning vector in step 3 and then stored in $\boldsymbol{Y}_{k-1}$ during step 4.
- The vector must have turned significantly from $\boldsymbol{Y}_{k-1}$ to $\overline{\boldsymbol{Y}}_k$.

8. Let $t_{k,2}$ denote the final value of $t_{k,2}$ obtained in step 7. Let $\boldsymbol{x}_i^{c,(k-1)}$ denote the i^{th} column of $\boldsymbol{X}_{k-1}^{c}$. Then, construct $\boldsymbol{Z}_{k-1} \in \mathbb{R}^{n \times r_k^c}$ by

$$\boldsymbol{Z}_{k-1} = \left[\boldsymbol{x}_1^{c,(k-1)} \quad \cdots \quad \boldsymbol{x}_{r_k^c - t_{k,2}}^{c,(k-1)} \;\middle|\; \boldsymbol{w}_1 \quad \cdots \quad \boldsymbol{w}_{t_{k,2}} \right] \tag{3.17}$$

Notes on step 8:

- The first $r_k^c - t_{k,2}$ columns of $\boldsymbol{Z}_{k-1}$ are copied from the first $r_k^c - t_{k,2}$ iteration vectors of $\boldsymbol{X}_{k-1}^{c}$.
- The last $t_{k,2}$ vectors in $\boldsymbol{Z}_{k-1}$ are the vectors from $\overline{\boldsymbol{Y}}_k$ that will become the additional turning vectors and new turning-of-turning vectors.

9. $\boldsymbol{M}$-orthogonalize $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{t_{k,2}}$. For $i = 1, \ldots, t_{k,2}$, do the following in (3.18) and

(3.19):

$$\tilde{\boldsymbol{y}}_i = \boldsymbol{w}_i - \underbrace{\boldsymbol{X}_{k-1}^a \left((\boldsymbol{X}_{k-1}^a)^T \boldsymbol{M} \boldsymbol{w}_i \right)}_{\substack{\boldsymbol{M}\text{-orthogonalize to} \\ \text{the vectors in } \boldsymbol{X}_{k-1}^a}} - \underbrace{\boldsymbol{\Phi}_{k-1} \left(\boldsymbol{\Phi}_{k-1}^T \boldsymbol{M} \boldsymbol{w}_i \right)}_{\substack{\boldsymbol{M}\text{-orthogonalize to} \\ \text{the vectors in } \boldsymbol{\Phi}_{k-1}}}$$
$$- \underbrace{\boldsymbol{Y}_{k-1} \left(\boldsymbol{Y}_{k-1}^T \boldsymbol{M} \boldsymbol{w}_i \right)}_{\substack{\boldsymbol{M}\text{-orthogonalize to} \\ \text{the vectors in } \boldsymbol{Y}_{k-1}}} - \underbrace{\sum_{j=1}^{r_k^c - t_{k,2} + i - 1} \boldsymbol{z}_j^{(k-1)} \left(\left(\boldsymbol{z}_j^{(k-1)} \right)^T \boldsymbol{M} \boldsymbol{w}_i \right)}_{\substack{\boldsymbol{M}\text{-orthogonalize to the first} \\ r_k^c - t_{k,2} \text{ vectors in } \boldsymbol{X}_{k-1}^c \text{ and} \\ \text{to the first } i-1 \text{ turning-of-turning vectors}}} \tag{3.18}$$

$$\boldsymbol{z}_{r_k^c - t_{k,2} + i}^{(k-1)} = \frac{\tilde{\boldsymbol{y}}_i}{\sqrt{\tilde{\boldsymbol{y}}_i^T \boldsymbol{M} \tilde{\boldsymbol{y}}_i}} \tag{3.19}$$

Notes on step 9:

- The vectors $\tilde{\boldsymbol{y}}_i$, and their normalizations in (3.19), are the additional turning vectors and new turning-of-turning vectors.
- The purpose of the $\boldsymbol{M}$-orthonormalization in this step is to obtain a more numerically stable basis for $\boldsymbol{Z}_{k-1}$, which is necessary since $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{t_{k,2}}$ can potentially be very parallel after the inverse iteration of step 6.

10. Perform simultaneous inverse iteration on $\boldsymbol{Z}_{k-1}$ to obtain $\overline{\boldsymbol{Z}}_k$:

$$\overline{\boldsymbol{Z}}_k = \boldsymbol{K}^{-1} \boldsymbol{M} \boldsymbol{Z}_{k-1}, \qquad \overline{\boldsymbol{Z}}_k \in \mathbb{R}^{n \times r_k^c} \tag{3.20}$$

Notes on step 10:

- The first $r_k^c - t_{k,2}$ vectors of $\overline{\boldsymbol{Z}}_k$ have now undergone inverse iteration for the first time in this iteration, which they would have done in the basic and enriched methods, as well. However, the last $t_{k,2}$ vectors of $\overline{\boldsymbol{Z}}_k$ contain a mixture of additional forward turning vectors and, importantly, forward turning-of-turning vectors.
- The new forward turning vectors are vectors that began this iteration in

$\boldsymbol{X}_{k-1}^{b}$, were not replaced during the construction of $\boldsymbol{Y}_{k-1}$, and were determined to have turned significantly from $\boldsymbol{Y}_{k-1}$ to $\overline{\boldsymbol{Y}}_k$. Similarly to the forward turning vectors computed in step 6 of the algorithm, these vectors have undergone inverse iteration twice during a single iteration of the algorithm.

- The new forward turning-of-turning vectors are vectors that began this iteration in $\boldsymbol{X}_{k-1}^{a}$, were selected as turning vectors in step 3 of the algorithm, and were determined to have turned significantly from $\boldsymbol{Y}_{k-1}$ to $\overline{\boldsymbol{Y}}_k$. These iteration vectors have undergone inverse iteration three times during a single iteration of the algorithm and are the key to the double enrichment of the E^2 method.

11. Define the new q-dimensional subspace:

$$E_k = \text{span}\left\{\overline{\boldsymbol{X}}_k\right\} \quad \text{where} \quad \overline{\boldsymbol{X}}_k = \left[\boldsymbol{\Phi}_{k-1} \quad \overline{\boldsymbol{X}}_k^a \quad \overline{\boldsymbol{Y}}_k \quad \overline{\boldsymbol{Z}}_k\right] \tag{3.21}$$

12. Project the stiffness and mass matrices onto the current subspace:

$$\begin{aligned} \boldsymbol{K}_k &= \overline{\boldsymbol{X}}_k^T \boldsymbol{K} \overline{\boldsymbol{X}}_k, \qquad & \boldsymbol{K}_k \in \mathbb{R}^{q\times q} \\ \boldsymbol{M}_k &= \overline{\boldsymbol{X}}_k^T \boldsymbol{M} \overline{\boldsymbol{X}}_k, \qquad & \boldsymbol{M}_k \in \mathbb{R}^{q\times q} \end{aligned} \tag{3.22}$$

13. Determine $\boldsymbol{Q}_k \in \mathbb{R}^{q\times q}$ and $\boldsymbol{\Omega}_k^2 \in \mathbb{R}^{q\times q}$ by solving the generalized eigenvalue problem of the projected operators:

$$\boldsymbol{K}_k \boldsymbol{Q}_k = \boldsymbol{M}_k \boldsymbol{Q}_k \boldsymbol{\Omega}_k^2 \tag{3.23}$$

14. Compute an improved approximation of the eigenvectors using the Ritz coordinates obtained in (3.23):

$$\boldsymbol{X}_k = \overline{\boldsymbol{X}}_k \boldsymbol{Q}_k \tag{3.24}$$

15. Use (3.25) to determine the convergence of the eigenvalues:

$$\left[1 - \frac{\left(\omega_i^{2,(k)}\right)^2}{\left(\boldsymbol{q}_i^{rr,(k)}\right)^T \boldsymbol{q}_i^{rr,(k)}}\right]^{1/2} \leq tolc, \quad i = p_{k-1}+1, \ldots, p, \tag{3.25}$$

where $\boldsymbol{q}_i^{rr,(k)}$ is the $(i - p_{k-1})^{\text{th}}$ column of $\boldsymbol{Q}_k^{rr}$, which is square matrix of order $\left(r_k^a + r_k^b + r_k^c\right)$. $\boldsymbol{Q}_k^{rr}$ defined by

$$\boldsymbol{Q}_k = \begin{bmatrix} \boldsymbol{Q}_k^{cc} & \boldsymbol{Q}_k^{cr} \\ \boldsymbol{Q}_k^{rc} & \boldsymbol{Q}_k^{rr} \end{bmatrix}. \tag{3.26}$$

16. Update the number of converged iteration vectors to p_k. If $p_k < p$, then return to step 1 to continue the iteration.

In the algorithm above, steps 1-6 are the same as the steps of the enriched subspace iteration method given in reference [18]. Steps 7-10 are specific to the E^2 method. Steps 11-16 are the same as the steps in the basic subspace iteration method. Generally, the algorithm above can be thought of as having five components:

1. Partition the iteration vectors (step 1).

2. Enrich the subspace by computing turning vectors and, ultimately, forward turning vectors (steps 2 - 6).

3. Further enrich the subspace by computing additional forward turning vectors and new forward turning-of-turning vectors (steps 7 - 10).

4. Perform Ritz analysis, which entails projecting the $\boldsymbol{K}$ and $\boldsymbol{M}$ operators onto the new subspace and solving the projected eigenvalue problem (steps 11 - 13).

5. Update the approximations of the iteration vectors using the results from the Ritz analysis and check for convergence (steps 14 - 16).

After convergence to E_∞, the Sturm sequence property is used to verify that the algorithm has not missed any of the p lowest eigenvalues. Let $\mu \in \mathbb{R}$ such that μ is slightly greater than ω_p^2. Then, factor the shifted matrix $\boldsymbol{K} - \mu\boldsymbol{M}$ into $\boldsymbol{LDL}^T$. The Sturm sequence property states that the number of negative elements on the diagonal of $\boldsymbol{D}$ is equal to the number of eigenvalues smaller than μ. Therefore, the results from our calculations are verified if there are exactly p negative elements in $\boldsymbol{D}$. If there are more than p negative elements in $\boldsymbol{D}$, this is an indication of one several possibilities:

1. The shift μ was too large and should be reduced to be closer to ω_p^2.

2. ω_p^2 has a multiplicity greater than 1.

3. One or more eigenpairs were missed during the subspace iteration. In this case, it is recommended to increase the dimensionality of the subspace by appending random vectors to the current set of iteration vectors. Then, the subspace iteration should be continued with the new iteration vectors.

3.1.2 Flow Chart Representations of the E^2 Algorithm

In this section, we visualize how the enrichment steps of the E^2 algorithm are integrated into the basic subspace iteration method. As indicated in Figure 3-1, the enrichment steps of the E^2 method are easily added to the basic method. In fact, the primary difference between the basic, enriched, and E^2 methods is with regard to how the new subspace is defined in each iteration. In the basic method, the new subspace is obtained by applying inverse iteration to all of the iteration vectors. In the enriched method, some iteration vectors are discarded and forward turning vectors are incorporated when defining the new subspace. In the E^2 method, some iteration vectors are discarded and forward turning vectors as well as forward turning-of-turning vectors are used when defining the new subspace.

In Figure 3-2, we visualize how the E^2 algorithm proceeds through the steps described in the previous section.

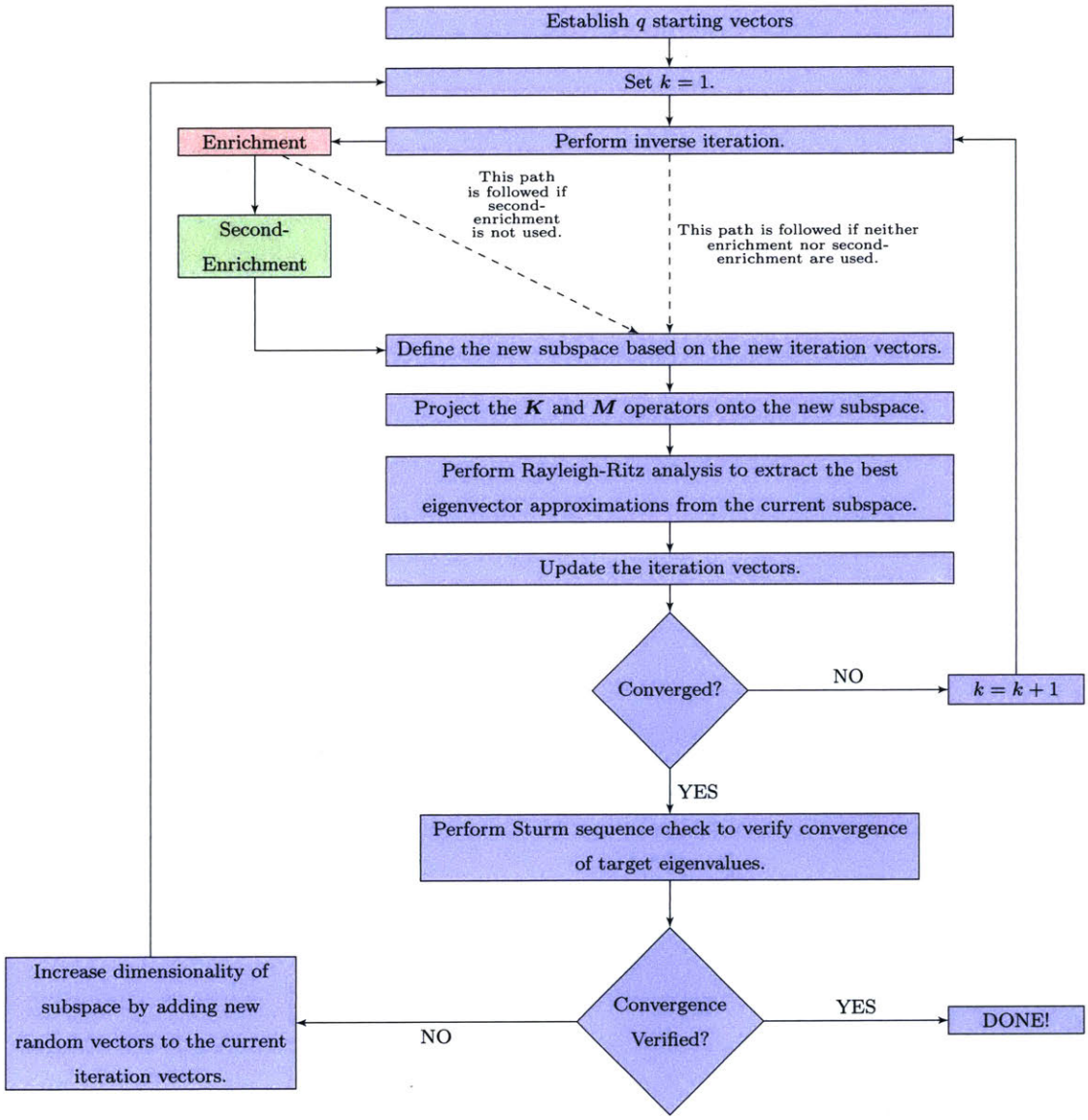

Figure 3-1: Flow chart depicting how the enrichment and second enrichment procedures are integrated into the basic subspace iteration method. The simple integration of these procedures makes them attractive acceleration schemes. The blue boxes show the steps of the basic method. The red box is the first enrichment step in which turning vectors are calculated and replace less effective iteration vectors. The green box is the second enrichment step in which turning-of-turning vectors are calculated and replace additional iteration vectors.

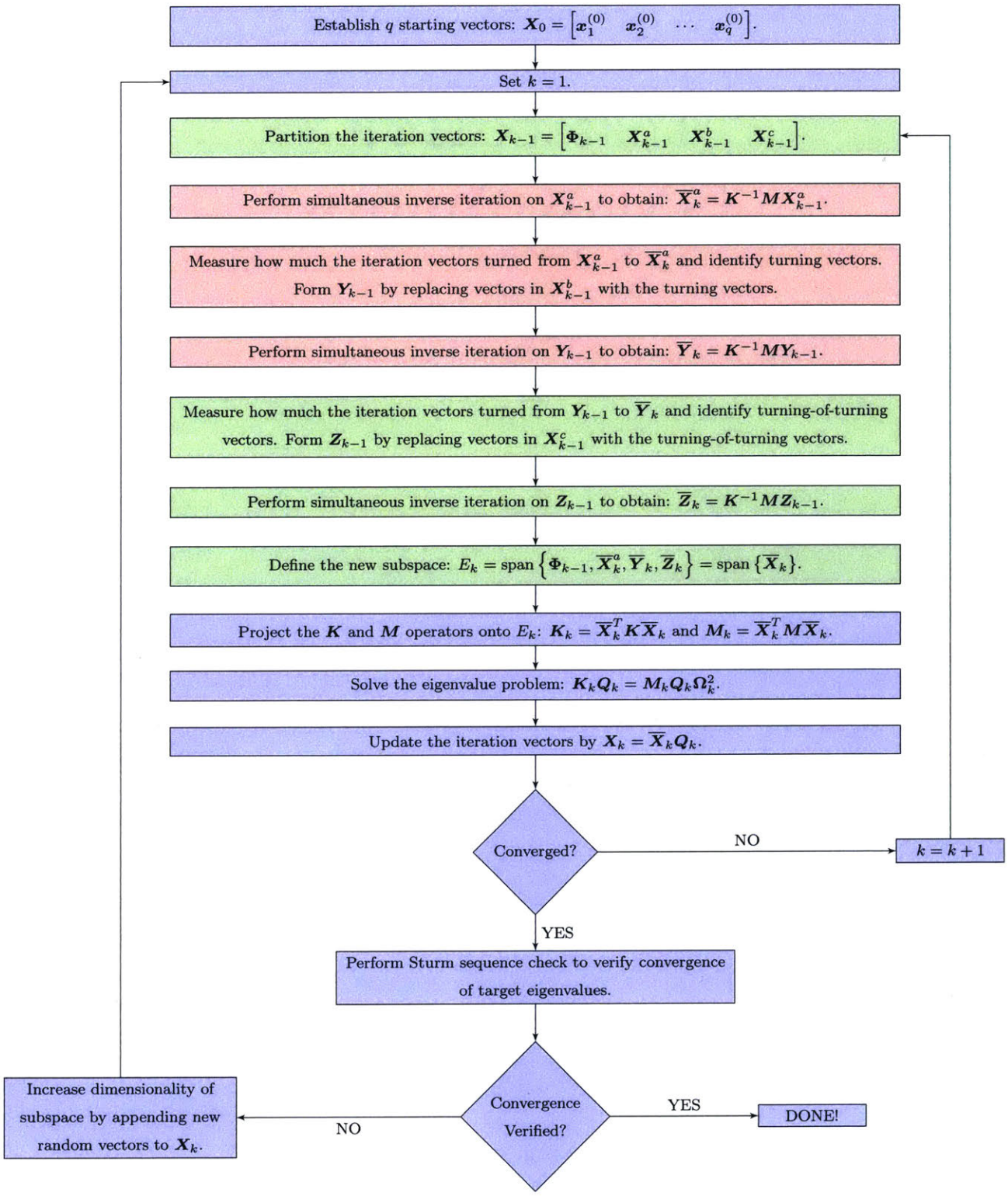

Figure 3-2: Algorithm flow chart for the E^2 subspace iteration method. The blue boxes indicate the steps of the E^2 method that are the same as in the basic method. The red boxes indicate the steps of the E^2 method that are the same as in the enriched method. The green boxes indicate the steps of this procedure that are unique to the E^2 method. Note: the partition step is highlighted in green, even though there is also a partition step in the enriched method because the enriched method does not partition iteration vectors into $\boldsymbol{X}_{k-1}^c$.

The Sturm sequence check is included in Figures 3-1 and 3-2 as an important step of the E^2 algorithm. The figures consider the situation in which the Sturm sequence check fails and it is necessary to continue the subspace iteration. It is important to note that while it is theoretically possible to fail to converge to one of the target eigenvectors, for example if all of the starting vectors are $\boldsymbol{M}$-orthogonal to that eigenvector, this situation rarely occurs in practice when Equation (3.5) is used to determine q.

3.1.3 Visualization of Turning Vectors and Turning-of-Turning Vectors

In this section, we illustrate the process by which turning vectors and turning-of-turning vectors are calculated. We also depict the relationship between turning vectors and the corresponding forward turning vectors, as well as the relationship between turning-of-turning vectors and the corresponding forward turning-of-turning vectors.

Both the basic and enriched subspace methods can be implemented using as few as two starting vectors. Hence, the subspaces E_{k-1} and E_k can be depicted in the smallest-dimension case as planes. In Figures 3-3 and 3-4, we provide an illustration of how these two methods define the new subspace, E_k, from the iteration vectors in the current subspace, E_{k-1}.

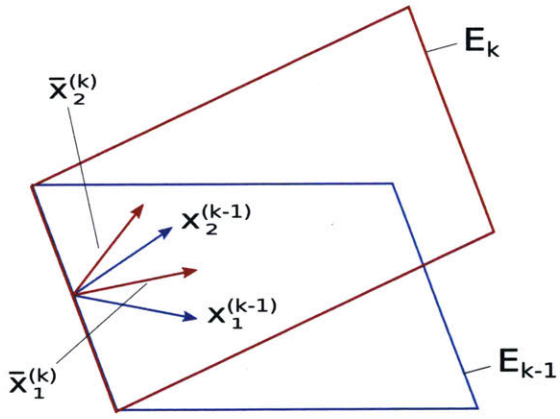

Figure 3-3: Geometric illustration of the subspaces E_{k-1} and E_k using the original Bathe subspace iteration method. In this illustration, $\boldsymbol{M} = \boldsymbol{I}$, and two iteration vectors are depicted. The depicted subspaces are defined by $E_{k-1} = \text{span}\left\{\boldsymbol{x}_1^{(k-1)}, \boldsymbol{x}_2^{(k-1)}\right\}$ and $E_k = \text{span}\left\{\overline{\boldsymbol{x}}_1^{(k)}, \overline{\boldsymbol{x}}_2^{(k)}\right\}$. The vectors $\overline{\boldsymbol{x}}_1^{(k)}$ and $\overline{\boldsymbol{x}}_2^{(k)}$ are obtained by applying inverse iteration to $\boldsymbol{x}_1^{(k-1)}$ and $\boldsymbol{x}_2^{(k-1)}$, respectively.

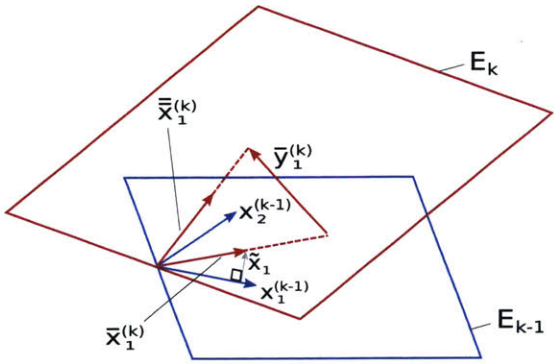

Figure 3-4: Geometric illustration of the subspaces E_{k-1} and E_k using the enriched Bathe subspace iteration method. In this illustration, $\boldsymbol{M} = \boldsymbol{I}$, and two iteration vectors are depicted. The depicted subspaces are defined by $E_{k-1} = \text{span}\left\{\boldsymbol{x}_1^{(k-1)}, \boldsymbol{x}_2^{(k-1)}\right\}$ and $E_k = \text{span}\left\{\bar{\boldsymbol{x}}_1^{(k)}, \bar{\boldsymbol{y}}_1^{(k)}\right\}$. The vector $\bar{\boldsymbol{x}}_1^{(k)}$ is obtained by applying inverse iteration to $\boldsymbol{x}_1^{(k-1)}$, and $\bar{\boldsymbol{y}}_1^{(k)}$ is the forward turning vector that is the key ingredient for the acceleration of the enriched method.

The acceleration of the enriched method is due to the fact that $\bar{\boldsymbol{y}}_1^{(k)}$ is a linear combination of $\bar{\boldsymbol{x}}_1^{(k)}$ and $\bar{\bar{\boldsymbol{x}}}_1^{(k)}$, which represents $\boldsymbol{x}_1^{(k-1)}$ after two inverse iterations. Therefore, $\bar{\bar{\boldsymbol{x}}}_1^{(k)}$ is in the new subspace that has been created using the forward turning vector, $E_k = \text{span}\left\{\bar{\boldsymbol{x}}_1^{(k)}, \bar{\boldsymbol{y}}_1^{(k)}\right\}$. Consequently, during the Ritz analysis step, either $\bar{\bar{\boldsymbol{x}}}_1^{(k)}$ or a superior approximation for $\boldsymbol{\phi}_1$ can be extracted from the subspace. Since $\bar{\boldsymbol{y}}_1^{(k)}$ points from $\bar{\boldsymbol{x}}_1^{(k)}$ to $\bar{\bar{\boldsymbol{x}}}_1^{(k)}$, it is referred to as a forward turning vector.

As opposed to the basic and enriched methods, which can work on as few as two iteration vectors, the E^2 method requires at least three iteration vectors: one iteration vector that undergoes the initial inverse iteration step, a second iteration vector that can be replaced by a turning vector, and a third iteration vector that can be replaced by a turning-of-turning vector. In Figure 3-5, we visualize how the E^2 method iterates from $E_{k-1} = \text{span}\left\{\boldsymbol{x}_1^{(k-1)}, \boldsymbol{x}_2^{(k-1)}, \boldsymbol{x}_3^{(k-1)}\right\}$ to $E_k = \text{span}\left\{\bar{\boldsymbol{x}}_1^{(k)}, \bar{\boldsymbol{y}}_1^{(k)}, \bar{\boldsymbol{z}}_1^{(k)}\right\}$.

The upper graphic of Figure 3-5 shows the result of applying inverse iteration on $\boldsymbol{x}_1^{(k-1)}$ to obtain $\bar{\boldsymbol{x}}_1^{(k)}$. Then, $\tilde{\boldsymbol{x}}_1$ is identified as the turning vector that represents the direction $\boldsymbol{x}_1^{(k-1)}$ turned towards the target subspace after undergoing inverse iteration. The turning vector, $\tilde{\boldsymbol{x}}_1$, is then normalized to obtain $\boldsymbol{y}_1^{(k-1)}$, which is the input to the lower graphic of Figure 3-5.

In the lower graphic of Figure 3-5, $\boldsymbol{y}_1^{(k-1)}$ undergoes inverse iteration to obtain $\bar{\boldsymbol{y}}_1^{(k)}$.

Then, we identify $\tilde{\boldsymbol{y}}_1$ as the turning-of-turning vector. The turning-of-turning vector is normalized and then undergoes inverse iteration to obtain the corresponding forward turning-of-turning vector, $\overline{\boldsymbol{z}}_1^{(k)}$. The acceleration of the E^2 method is due to the fact that $\overline{\boldsymbol{z}}_1^{(k)}$ is a linear combination of $\overline{\boldsymbol{y}}_1^{(k)}$ and $\overline{\overline{\overline{\boldsymbol{x}}}}_1^{(k)}$, which represents $\boldsymbol{x}_1^{(k-1)}$ after undergoing three inverse iterations. Therefore, $\overline{\overline{\overline{\boldsymbol{x}}}}_1^{(k)}$ is in the new subspace that has been created using the forward turning vector and forward turning-of-turning vector. Consequently, during the Ritz analysis step, either $\overline{\overline{\overline{\boldsymbol{x}}}}_1^{(k)}$ or a superior approximation for $\boldsymbol{\phi}_1$ can be extracted from the subspace. Since $\overline{\boldsymbol{z}}_1^{(k)}$ points from $\overline{\boldsymbol{y}}_1^{(k)}$ (recall that $\overline{\boldsymbol{y}}_1^{(k)}$ is the normalization of the turning vector identified in the upper graphic of Figure 3-5) to $\overline{\overline{\overline{\boldsymbol{x}}}}_1^{(k)}$, it is referred to as a forward turning-of-turning vector.

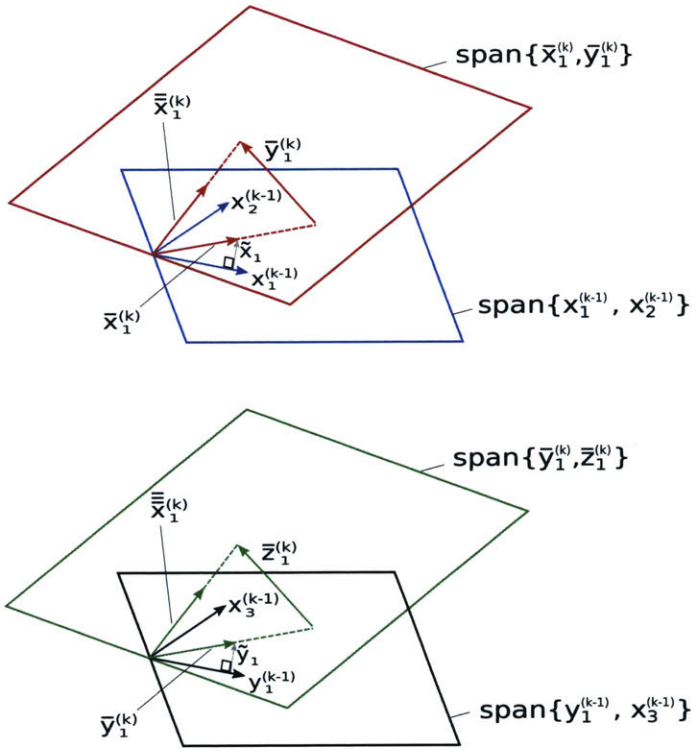

Figure 3-5: Geometric illustration of the E^2 Bathe subspace iteration method showing the computation of the inverse iteration step on $\boldsymbol{x}_1^{(k-1)}$, which results in $\overline{\boldsymbol{x}}_1^{(k)}$; the computation of the forward turning vector, $\overline{\boldsymbol{y}}_1^{(k)}$; as well as the computation of the forward turning-of-turning vector, $\overline{\boldsymbol{z}}_1^{(k)}$, which is the key ingredient for the additional enrichment of the E^2 method. In this illustration, $\boldsymbol{M} = \boldsymbol{I}$, and three iteration vectors are depicted. In the E^2 method, the previous and current subspaces are defined as $E_{k-1} = \text{span}\left\{\boldsymbol{x}_1^{(k-1)}, \boldsymbol{x}_2^{(k-1)}, \boldsymbol{x}_3^{(k-1)}\right\}$ and $E_k = \text{span}\left\{\overline{\boldsymbol{x}}_1^{(k)}, \overline{\boldsymbol{y}}_1^{(k)}, \overline{\boldsymbol{z}}_1^{(k)}\right\}$, respectively.

In Figure 3-5, observe that we have shown $\overline{\overline{\boldsymbol{x}}}_1^{(k)}$ and $\overline{\overline{\overline{\boldsymbol{x}}}}_1^{(k)}$. These vectors represent

$\boldsymbol{x}_1^{(k-1)}$ after two and three inverse iterations, respectively. Hence,

$$\begin{aligned}
\overline{\boldsymbol{x}}_1^{(k)} &= \boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{x}_1^{(k-1)} \\
\overline{\overline{\boldsymbol{x}}}_1^{(k)} &= \boldsymbol{K}^{-1}\boldsymbol{M}\overline{\boldsymbol{x}}_1^{(k)} \\
\overline{\overline{\overline{\boldsymbol{x}}}}_1^{(k)} &= \boldsymbol{K}^{-1}\boldsymbol{M}\overline{\overline{\boldsymbol{x}}}_1^{(k)}
\end{aligned} \tag{3.27}$$

First, note that neither $\overline{\overline{\boldsymbol{x}}}_1^{(k)}$ nor $\overline{\overline{\overline{\boldsymbol{x}}}}_1^{(k)}$ are actually computed in the E^2 method. However, they are shown in Figure 3-5 to illustrate the following observations:

1. The forward turning vector, $\overline{\boldsymbol{y}}_1^{(k)}$, points from the space spanned by $\overline{\boldsymbol{x}}_1^{(k)}$ to the space that would be spanned by $\overline{\boldsymbol{x}}_1^{(k)}$ after undergoing an additional step of inverse iteration. That is, $\overline{\boldsymbol{y}}_1^{(k)}$ is a linear combination of $\overline{\boldsymbol{x}}_1^{(k)}$ and $\overline{\overline{\boldsymbol{x}}}_1^{(k)}$. This is the reason why we refer to $\overline{\boldsymbol{y}}_1^{(k)}$ as a forward turning vector.

2. The forward turning-of-turning vector, $\overline{\boldsymbol{z}}_1^{(k)}$, points from the space spanned by $\overline{\boldsymbol{y}}_1^{(k)}$ to the space that would be spanned by $\overline{\boldsymbol{y}}_1^{(k)}$ after an additional step of inverse iteration. That is, $\overline{\boldsymbol{z}}_1^{(k)}$ is a linear combination of $\overline{\boldsymbol{y}}_1^{(k)}$ and $\overline{\overline{\overline{\boldsymbol{x}}}}_1^{(k)}$. This is the reason why we refer to $\overline{\boldsymbol{z}}_1^{(k)}$ as a forward turning-of-turning vector.

To demonstrate the turning process, consider the following simple example. Define $\boldsymbol{K}$, $\boldsymbol{M}$, and $\boldsymbol{x}_{k-1}$ as follows:

$$\boldsymbol{K} = \begin{bmatrix} 1 & 2 \\ 2 & 8 \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{x}_{k-1} = \frac{1}{\sqrt{22}} \begin{bmatrix} \sqrt{17} \\ \sqrt{5} \end{bmatrix} \approx \begin{bmatrix} 0.879049 \\ 0.476731 \end{bmatrix} \tag{3.28}$$

Note that $\boldsymbol{x}_{k-1}$ is $\boldsymbol{M}$-orthonormal. Then, $\overline{\boldsymbol{x}}_k$ is obtained by inverse iteration as follows:

$$\begin{aligned}
\overline{\boldsymbol{x}}_k = \boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{x}_{k-1} &= \frac{1}{\sqrt{22}} \begin{bmatrix} 2 & -1/2 \\ -1/2 & 1/4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{17} \\ \sqrt{5} \end{bmatrix} \\
&= \frac{1}{\sqrt{22}} \begin{bmatrix} 2\sqrt{17} - \sqrt{5}/2 \\ -\sqrt{17}/2 + \sqrt{5}/5 \end{bmatrix} \\
&\approx \begin{bmatrix} 1.51973 \\ -0.320342 \end{bmatrix}
\end{aligned} \tag{3.29}$$

The vectors $\boldsymbol{x}_{k-1}$ and $\overline{\boldsymbol{x}}_k$ are depicted in Figure 3-6 as the solid blue and red arrows, respectively. The turning vector is found by projecting $\overline{\boldsymbol{x}}_k$ onto the subspace spanned by $\boldsymbol{x}_{k-1}$. This process is Gram-Schmidt orthogonalization, which is described in reference [5]. The projection is denoted $\text{proj}_{\boldsymbol{x}_{k-1}}(\overline{\boldsymbol{x}}_k)$ and is computed by

$$\text{proj}_{\boldsymbol{x}_{k-1}}(\overline{\boldsymbol{x}}_k) = \boldsymbol{x}_{k-1}\frac{\boldsymbol{x}_{k-1}^T\boldsymbol{M}\overline{\boldsymbol{x}}_k}{\boldsymbol{x}_{k-1}^T\boldsymbol{M}\boldsymbol{x}_{k-1}} = \boldsymbol{x}_{k-1}\boldsymbol{x}_{k-1}^T\boldsymbol{M}\overline{\boldsymbol{x}}_k \tag{3.30}$$

In (3.30), we have invoked the $\boldsymbol{M}$-orthonormality of $\boldsymbol{x}_{k-1}$, and so $\boldsymbol{x}_{k-1}^T\boldsymbol{M}\boldsymbol{x}_{k-1} = 1$. The projection of $\overline{\boldsymbol{x}}_k$ onto the subspace spanned by $\boldsymbol{x}_{k-1}$ is depicted as a dashed red line in Figure 3-6. Once the projection is computed, then the turning vector is computed by

$$\begin{aligned}\text{Turning Vector} = \overline{\boldsymbol{x}}_k - \text{proj}_{\boldsymbol{x}_{k-1}}(\overline{\boldsymbol{x}}_k) &\approx \begin{bmatrix}1.51973\\-0.320342\end{bmatrix} - \begin{bmatrix}1.04009\\0.56407\end{bmatrix} \\ &\approx \begin{bmatrix}0.479639\\-0.884411\end{bmatrix}\end{aligned} \tag{3.31}$$

The turning vector that is visualized in Figure 3-6 is obtained by translating the vector in (3.31) such that the tail of the vector in (3.31) is located at the tip of $\text{proj}_{\boldsymbol{x}_{k-1}}(\overline{\boldsymbol{x}}_k)$.

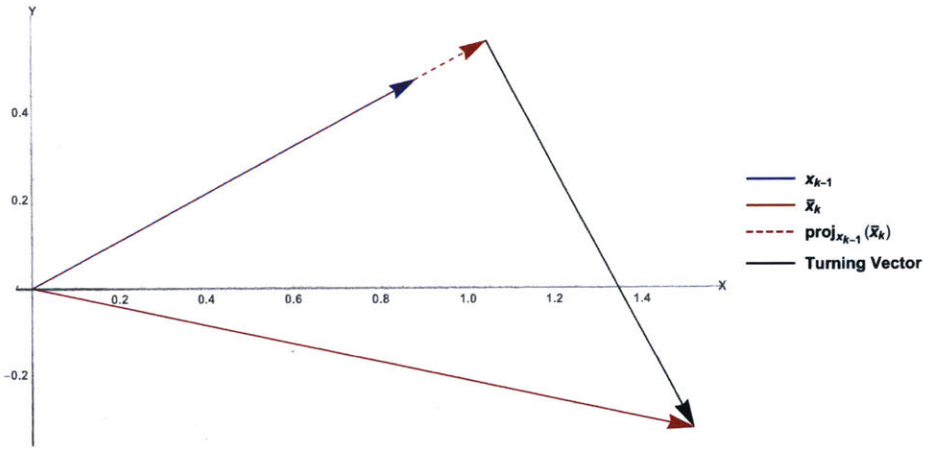

Figure 3-6: Illustration of vector projection and the calculation of a turning vector using the Gram-Schmidt orthogonalization algorithm. $\overline{\boldsymbol{x}}_k$ is projected onto the subspace spanned by $\boldsymbol{x}_{k-1}$. Then, the turning vector is defined as the difference between $\overline{\boldsymbol{x}}_k$ and its projection onto $\boldsymbol{x}_{k-1}$. To visualize the relationship between $\text{proj}_{\boldsymbol{x}_{k-1}}(\overline{\boldsymbol{x}}_k)$ and $\overline{\boldsymbol{x}}_k$, the turning vector has been translated so that the tail of the turning vector is located at the tip of $\text{proj}_{\boldsymbol{x}_{k-1}}(\overline{\boldsymbol{x}}_k)$.

3.2 Tabulation of Floating Point Operations

In this section, we report the total number of floating point operations used by the basic subspace iteration method, the enriched subspace iteration method, and the E^2 subspace iteration method. The results of Tables 3.2 and 3.3 can be found in references [5] and [18], respectively, and are only repeated here for reference and comparison to the new table for the E^2 method. Table 3.4 contains an accounting of the number of floating point operations used in the E^2 method and is original to this thesis. In Table 3.3, the iteration vectors were assumed to be partitioned equally, hence $r_k^a = r_k^b$, and we denoted this partition size by r_k. In Table 3.4, we have also assumed that the iteration vectors are partitioned equally, so $r_k^a = r_k^b = r_k^c$, and we denoted this partition size by r'_k.

Symbol	Meaning
$r_k = \frac{q-p_{k-1}}{2}$	The number of non-converged iteration vectors used in the enriched method
$r'_k = \frac{q-p_{k-1}}{3}$	The number of non-converged iteration vectors used in the E^2 method
$t_{k,1}$	The number of turning vectors identified during the enrichment step of the enriched method
$t'_{k,1}$	The number of turning vectors identified during the first enrichment step of the E^2 method
$t_{k,2}$	The number of additional turning vectors and new turning-of-turning vectors identified during the second enrichment step of the E^2 method
$\boldsymbol{R}_{k-1} = \boldsymbol{M}\boldsymbol{X}_{k-1}$	The iteration vectors multiplied by $\boldsymbol{M}$
$\boldsymbol{\Psi}_{k-1} = \boldsymbol{M}\boldsymbol{\Phi}_{k-1}$	The converged iteration vectors multiplied by $\boldsymbol{M}$
m	The mean half-bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$

Table 3.1: List of symbols and their associated meanings used in Tables 3.2 - 3.4.

Basic Subspace Iteration Method		
Operation	Calculation	Number of operations
Factorization of $\boldsymbol{K}$	$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$	$\frac{1}{2}nm^2 + \frac{3}{2}nm$
Define $\boldsymbol{R}_0$	$\boldsymbol{R}_0 = \boldsymbol{M}\boldsymbol{X}_0$	
For $k = 1, 2, \ldots,$ repeat steps (a) - (f):		
(a) Inverse iteration	$\overline{\boldsymbol{X}}_k = \boldsymbol{K}^{-1}\boldsymbol{R}_{k-1}$	$nq(2m+1)$
(b) Project the $\boldsymbol{K}$ Operator	$\boldsymbol{K}_k = \overline{\boldsymbol{X}}_k^T\boldsymbol{R}_{k-1}$	$\frac{1}{2}nq(q+1)$
(c) Compute $\overline{\boldsymbol{R}}_k$	$\overline{\boldsymbol{R}}_k = \boldsymbol{M}\overline{\boldsymbol{X}}_k$	$nq(2m+1)$
(d) Project the $\boldsymbol{M}$ Operator	$\boldsymbol{M}_k = \overline{\boldsymbol{X}}_k^T\overline{\boldsymbol{R}}_k$	$\frac{1}{2}nq(q+1)$
(e) Solve the eigenproblem	$\boldsymbol{K}_k\boldsymbol{Q}_k = \boldsymbol{M}_k\boldsymbol{Q}_k\boldsymbol{\Omega}_k^2$	$\mathcal{O}\left(q^3\right)$
(f) Update iteration vectors	$\boldsymbol{R}_k = \overline{\boldsymbol{R}}_k\boldsymbol{Q}_k$	nq^2
Total in a single iteration of (a)-(f):		$nq(4m+q+3) + nq^2 + \mathcal{O}\left(q^3\right)$
Sturm sequence check	$\overline{\boldsymbol{K}} = \boldsymbol{K} - \mu\boldsymbol{M}$	$n(m+1)$
	$\overline{\boldsymbol{K}} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$	$\frac{1}{2}nm^2 + \frac{3}{2}nm$

Table 3.2: Tabulation of floating point operations for the basic Bathe subspace iteration method when the algorithm iterates on all iteration vectors, including converged iteration vectors.

Enriched Subspace Iteration Method		
Operation	Calculation	Number of operations
Factorization of $\boldsymbol{K}$	$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$	$\frac{1}{2}nm^2 + \frac{3}{2}nm$
Define $\boldsymbol{R}_0$	$\boldsymbol{R}_0 = \boldsymbol{M}\boldsymbol{X}_0$	
For $k = 1, 2, \ldots,$ repeat steps (a) - (i):		
(a) Partition iteration vectors	$\boldsymbol{R}_{k-1} = \begin{bmatrix}\boldsymbol{\Psi}_{k-1} & \boldsymbol{R}_{k-1}^a & \boldsymbol{R}_{k-1}^b\end{bmatrix}$	
(b) Inverse iteration	$\overline{\boldsymbol{X}}_k^a = \boldsymbol{K}^{-1}\boldsymbol{R}_{k-1}^a$	$nr_k(2m+1)$
(c) Compute $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k$	$\begin{bmatrix}\boldsymbol{D}_k & \boldsymbol{A}_k & \boldsymbol{B}_k\end{bmatrix} = \left(\overline{\boldsymbol{X}}_k^a\right)^T\boldsymbol{R}_{k-1}$	$nr_k\left(p_{k-1} + \frac{r_k+1}{2} + r_k\right)$
	$\overline{\boldsymbol{R}}_k^a = \boldsymbol{M}\overline{\boldsymbol{X}}_k^a$	$nr_k(2m+1)$
	$\boldsymbol{C}_k = \left(\overline{\boldsymbol{X}}_k^a\right)^T\overline{\boldsymbol{R}}_k^a$	$\frac{1}{2}nr_k(r_k+1)$
(d) Check the amount of turning	See algorithm	$\mathcal{O}\left(qr_k^2\right)$
(e) Calculate $\boldsymbol{S}_{k-1} = \boldsymbol{M}\boldsymbol{Y}_{k-1}$	See algorithm	$nt_{k,1}(q+1)$
(f) Inverse iteration on turning vectors	$\overline{\boldsymbol{Y}}_k = \boldsymbol{K}^{-1}\boldsymbol{S}_{k-1}$	$nr_k(2m+1)$
	$\overline{\boldsymbol{X}}_{k+1} = \begin{bmatrix}\boldsymbol{\Phi}_{k-1} & \overline{\boldsymbol{X}}_k^a & \overline{\boldsymbol{Y}}_k\end{bmatrix}$	
(g) Project the $\boldsymbol{K}$ and $\boldsymbol{M}$ operators	$\boldsymbol{K}_k = \begin{bmatrix}\boldsymbol{\Omega}_{k-1}^2 & & \text{sym.} \\ \boldsymbol{0} & \boldsymbol{A}_k & \\ \boldsymbol{0} & \overline{\boldsymbol{Y}}_k^T\boldsymbol{R}_{k-1}^a & \overline{\boldsymbol{Y}}_k^T\boldsymbol{S}_{k-1}\end{bmatrix}$	$nr_k\left(t_{k,1} + \frac{r_k+1}{2}\right)$
	$\overline{\boldsymbol{R}}_k^b = \boldsymbol{M}\overline{\boldsymbol{Y}}_k$	$nr_k(2m+1)$
	$\boldsymbol{M}_k = \begin{bmatrix}\boldsymbol{I} & & \text{sym.} \\ \boldsymbol{D}_k^T & \boldsymbol{C}_k & \\ \overline{\boldsymbol{Y}}_k^T\boldsymbol{\Psi}_{k-1} & \overline{\boldsymbol{Y}}_k^T\overline{\boldsymbol{R}}_k^a & \overline{\boldsymbol{Y}}_k^T\overline{\boldsymbol{R}}_k^b\end{bmatrix}$	$nr_k\left(p_{k-1} + \frac{r_k+1}{2} + r_k\right)$
(h) Solve the eigenproblem	$\boldsymbol{K}_k\boldsymbol{Q}_k = \boldsymbol{M}_k\boldsymbol{Q}_k\boldsymbol{\Omega}_k^2$	$\mathcal{O}\left(q^3\right)$
(i) Update iteration vectors	$\boldsymbol{R}_k = \begin{bmatrix}\boldsymbol{\Psi}_{k-1} & \overline{\boldsymbol{R}}_k^a & \overline{\boldsymbol{R}}_k^b\end{bmatrix}\boldsymbol{Q}_k$	nq^2
Total in a single iteration of (a) - (i):		$2nr_k(4m+q+3) + nq^2 + nt_{k,1}(q+r_k+1) + \mathcal{O}\left(q^3\right)$
Sturm sequence check	$\overline{\boldsymbol{K}} = \boldsymbol{K} - \mu\boldsymbol{M}$	$n(m+1)$
	$\overline{\boldsymbol{K}} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$	$\frac{1}{2}nm^2 + \frac{3}{2}nm$

Table 3.3: Tabulation of floating point operations for the enriched Bathe subspace iteration method.

E^2 Subspace Iteration Method

Operation	Calculation	Number of operations
Factorization of $\boldsymbol{K}$	$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$	$\frac{1}{2}nm^2 + \frac{3}{2}nm$
Define $\boldsymbol{R}_0$	$\boldsymbol{R}_0 = \boldsymbol{M}\boldsymbol{X}_0$	
For $k = 1, 2, \ldots,$ repeat steps (a) - (m):		
(a) Partition iteration vectors	$\boldsymbol{R}_{k-1} = \begin{bmatrix} \boldsymbol{\Psi}_{k-1} & \boldsymbol{R}^a_{k-1} & \boldsymbol{R}^b_{k-1} & \boldsymbol{R}^c_{k-1} \end{bmatrix}$	
(b) Inverse iteration	$\overline{\boldsymbol{X}}^a_k = \boldsymbol{K}^{-1}\boldsymbol{R}^a_{k-1}$	$nr'_k(2m+1)$
	$\overline{\boldsymbol{R}}^a_k = \boldsymbol{M}\overline{\boldsymbol{X}}^a_k$	$nr'_k(2m+1)$
(c) Compute $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k, \boldsymbol{F}_k$	$\begin{bmatrix} \boldsymbol{D}_k & \boldsymbol{A}_k & \boldsymbol{B}_k & \boldsymbol{C}_k \end{bmatrix} = \left(\overline{\boldsymbol{X}}^a_k\right)^T \boldsymbol{R}_{k-1}$	$nr'_k\left(p_{k-1} + \frac{r'_k+1}{2} + 2r'_k\right)$
	$\boldsymbol{F}_k = \left(\overline{\boldsymbol{X}}^a_k\right)^T \overline{\boldsymbol{R}}^a_k$	$\frac{1}{2}nr'_k(r'_k+1)$
(d) Check the amount of turning	See algorithm	$\mathcal{O}\left(q \cdot (r'_k)^2\right)$
(e) Calculate $\boldsymbol{S}_{k-1} = \boldsymbol{M}\boldsymbol{Y}_{k-1}$	See algorithm	$nt'_{k,1}(q+1)$
(f) Inverse iteration on turning vectors	$\overline{\boldsymbol{Y}}_k = \boldsymbol{K}^{-1}\boldsymbol{S}_{k-1}$	$nr'_k(2m+1)$
	$\overline{\boldsymbol{R}}^b_k = \boldsymbol{M}\overline{\boldsymbol{Y}}_k$	$nr'_k(2m+1)$
(g) Compute $\boldsymbol{A}'_k, \boldsymbol{B}'_k, \boldsymbol{C}'_k, \boldsymbol{D}'_k, \boldsymbol{F}'_k$	$\begin{bmatrix} \boldsymbol{D}'_k & \boldsymbol{A}'_k & \boldsymbol{B}'_k & \boldsymbol{C}'_k \end{bmatrix}$	
	$= \overline{\boldsymbol{Y}}^T_k \begin{bmatrix} \boldsymbol{\Psi}_{k-1} & \boldsymbol{R}^a_{k-1} & \boldsymbol{S}_{k-1} & \boldsymbol{R}^c_{k-1} \end{bmatrix}$	$nr'_k\left(p_{k-1} + \frac{r'_k+1}{2} + t'_{k,1} + r'_k\right)$
	$\boldsymbol{F}'_k = \left(\overline{\boldsymbol{Y}}_k\right)^T \overline{\boldsymbol{R}}^b_k$	$\frac{1}{2}nr'_k(r'_k+1)$
(h) Check the amount of turning-of-turning	See algorithm	$\mathcal{O}\left(q \cdot (r'_k)^2\right)$
(i) Calculate $\boldsymbol{T}_{k-1} = \boldsymbol{M}\boldsymbol{Z}_{k-1}$	See algorithm	$nt_{k,2}(q+1)$
(j) Inverse iteration on turning-of-turning vectors	$\overline{\boldsymbol{Z}}_k = \boldsymbol{K}^{-1}\boldsymbol{T}_{k-1}$	$nr'_k(2m+1)$
	$\overline{\boldsymbol{R}}^c_k = \boldsymbol{M}\overline{\boldsymbol{Z}}_k$	$nr'_k(2m+1)$
(k) Project the $\boldsymbol{K}$ and $\boldsymbol{M}$ operators	$\boldsymbol{K}_k = \begin{bmatrix} \boldsymbol{\Omega}^2_{k-1} & & & \text{sym.} \\ \boldsymbol{0} & \boldsymbol{A}_k & & \\ \boldsymbol{0} & \boldsymbol{A}'_k & \boldsymbol{B}'_k & \\ \boldsymbol{0} & \boldsymbol{Z}^T_{k-1}\overline{\boldsymbol{R}}^a_k & \boldsymbol{Z}^T_{k-1}\overline{\boldsymbol{R}}^b_k & \boldsymbol{Z}^T_{k-1}\overline{\boldsymbol{R}}^c_k \end{bmatrix}$	$nr'_k\left(2t_{k,2} + \frac{r'_k+1}{2}\right)$
	$\boldsymbol{M}_k = \begin{bmatrix} \boldsymbol{I} & & & \text{sym.} \\ \boldsymbol{D}^T_k & \boldsymbol{F}_k & & \\ (\boldsymbol{D}'_k)^T & \overline{\boldsymbol{Y}}^T_k\overline{\boldsymbol{R}}^a_k & \boldsymbol{F}'_k & \\ \overline{\boldsymbol{Z}}^T_k\boldsymbol{\Psi}_{k-1} & \overline{\boldsymbol{Z}}^T_k\overline{\boldsymbol{R}}^a_k & \overline{\boldsymbol{Z}}^T_k\overline{\boldsymbol{R}}^b_k & \overline{\boldsymbol{Z}}^T_k\overline{\boldsymbol{R}}^c_k \end{bmatrix}$	$nr'_k\left(p_{k-1} + \frac{r'_k+1}{2} + 3r'_k\right)$
(l) Solve the eigenproblem	$\boldsymbol{K}_k\boldsymbol{Q}_k = \boldsymbol{M}_k\boldsymbol{Q}_k\boldsymbol{\Omega}^2_k$	$\mathcal{O}\left(q^3\right)$
(m) Update iteration vectors	$\boldsymbol{R}_k = \begin{bmatrix} \boldsymbol{\Psi}_{k-1} & \overline{\boldsymbol{R}}^a_k & \overline{\boldsymbol{R}}^b_k & \overline{\boldsymbol{R}}^c_k \end{bmatrix} \boldsymbol{Q}_k$	nq^2
Total in a single iteration of (a) - (m):	$3nr'_k(4m+q+3) + nq^2 + nt'_{k,1}(q+r'_k+1) + nt_{k,2}(q+2r'_k+1) + \mathcal{O}\left(q^3\right)$	
Sturm sequence check	$\overline{\boldsymbol{K}} = \boldsymbol{K} - \mu\boldsymbol{M}$	$n(m+1)$
	$\overline{\boldsymbol{K}} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$	$\frac{1}{2}nm^2 + \frac{3}{2}nm$

Table 3.4: Tabulation of floating point operations for the E^2 Bathe subspace iteration method.

The following three observations pertain to the E^2 method and are used to avoid unnecessarily performing the same computations twice.

1. The first $r'_k - t'_{k,1}$ rows of $\boldsymbol{A}'_k$ are the first $r'_k - t'_{k,1}$ rows of $\boldsymbol{B}_k^T$.

2. The first $r'_k - t_{k,2}$ rows of $\boldsymbol{Z}_{k-1}^T \overline{\boldsymbol{R}}_k^a$ are the first $r'_k - t_{k,2}$ rows of $\boldsymbol{C}_k^T$.

3. The first $r'_k - t_{k,2}$ rows of $\boldsymbol{Z}_{k-1}^T \overline{\boldsymbol{R}}_k^b$ are the first $r'_k - t_{k,2}$ rows of $(\boldsymbol{C}'_k)^T$.

We conclude this section with a remark on the relationship between the number of floating point operations used by the enriched method and the number of floating point operations used by the E^2 method. Specifically, consider the situation in which we perform the two methods without ever accepting converged iteration vectors. Therefore, $p_{k-1} = 0$ for all $k = 1, 2, \ldots$, and the total number of iteration vectors being used in each iteration is always q. In this special case, the only difference between the enriched and the E^2 methods is that the E^2 method requires the computation of the turning-of-turning vectors.

To be specific, let $\mathcal{N}_0$, $\mathcal{N}_1$, and $\mathcal{N}_2$ denote the number of floating point operations in the basic method, enriched, and E^2 methods, respectively. We will analyze the special case when $p_{k-1} = 0$ for all $k = 1, 2, \ldots,$. In this case,

$$\begin{aligned}
\mathcal{N}_0 &= nq(4m + 2q + 3) + \mathcal{O}\left(q^3\right) \\
\mathcal{N}_1 &= nq(4m + 2q + 3) + nt_{k,1} + \frac{3nqt_{k,1}}{2} + \mathcal{O}\left(q^3\right) \\
\mathcal{N}_2 &= nq(4m + 2q + 3) + \frac{4nqt'_{k,1}}{3} + nt'_{k,1} + \frac{5nqt_{k,2}}{3} + nt_{k,2} + \mathcal{O}\left(q^3\right)
\end{aligned} \tag{3.32}$$

Then, assuming that the enriched and E^2 methods use the same number of turning vectors in a given iteration, that is $t_{k,1} = t'_{k,1}$, the following equation holds:

$$\mathcal{N}_2 - \mathcal{N}_1 = \frac{5nqt_{k,2}}{3} + nt_{k,2} - \frac{nqt'_{k,1}}{6} + \mathcal{O}\left(q^3\right) \tag{3.33}$$

Equation (3.33) represents the number of additional floating point operations performed by the E^2 method compared to the enriched method in a given iteration when both

methods use the same number of turning vectors. Therefore, (3.33) represents the cost of performing the second enrichment procedure of the E^2 method.

If we also assume that all possible turning-of-turning vectors are used, hence, $t'_{k,1} = t_{k,2}$, then the number of additional floating point operations of the E^2 method compared to the enriched method is:

$$\mathscr{N}_2 - \mathscr{N}_1 = \frac{3nqt_{k,2}}{2} + nt_{k,2} + \mathcal{O}\left(q^3\right) \tag{3.34}$$

In (3.34), we have assumed $t_{k,1} = t'_{k,1} = t_{k,2}$. In this case, the number of additional floating point operations of the E^2 method compared to the enriched method is the same, up to $\mathcal{O}\left(q^3\right)$, as the number of additional floating point operations of the enriched method compared to the basic method:

$$\begin{aligned} \mathscr{N}_1 - \mathscr{N}_0 &= \frac{3nqt_{k,1}}{2} + nt_{k,1} + \mathcal{O}\left(q^3\right) \\ &= \frac{3nqt_{k,2}}{2} + nt_{k,2} + \mathcal{O}\left(q^3\right) \\ &= \mathscr{N}_2 - \mathscr{N}_1 + \mathcal{O}\left(q^3\right) \end{aligned} \tag{3.35}$$

In reference [18], the author remarked that the enriched method is a relatively inexpensive addition to the basic method. Based on equations (3.33), (3.34) and (3.35), we conclude that the E^2 method is a relatively inexpensive addition to the enriched method.

3.3 A Simplified Convergence Analysis

In this section, we examine the effects of using both turning vectors and turning-of-turning vectors to enrich the subspace of each iteration in the E^2 algorithm. The goal of this analysis is to provide insight into how these vectors are used to enrich the subspace and actually accelerate the convergence of the iteration vectors to the target subspace.

As before, suppose $\boldsymbol{K}$ and $\boldsymbol{M}$ are symmetric, positive definite matrices of order n. Recall, we are interested in finding the lowest p generalized eigenvalues satisfying $\boldsymbol{K}\boldsymbol{\phi} = \omega^2\boldsymbol{M}\boldsymbol{\phi}$. Following the convention used in this thesis, the eigenvalues and corresponding

eigenvectors are ordered as follows:

$$\begin{aligned} &0 < \omega_1^2 \leq \omega_2^2 \leq \cdots \leq \omega_n^2 < \infty \\ &\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_n \end{aligned} \tag{3.36}$$

Let $\boldsymbol{X}_{k-1} \in \mathbb{R}^{n \times q}$, $p < q \leq n$, denote the iteration vectors at the beginning of the k^{th} iteration. The first step of the main loop in the algorithm is to partition the iteration vectors as follows:

$$\begin{aligned} \boldsymbol{X}_{k-1} = \begin{bmatrix} \boldsymbol{X}_{k-1}^a & \boldsymbol{X}_{k-1}^b & \boldsymbol{X}_{k-1}^c \end{bmatrix} \qquad & \boldsymbol{X}_{k-1}^a \in \mathbb{R}^{n \times r_k^a}, \quad \boldsymbol{X}_{k-1}^b \in \mathbb{R}^{n \times r_k^b}, \\ & \boldsymbol{X}_{k-1}^c \in \mathbb{R}^{n \times r_k^c} \end{aligned} \tag{3.37}$$

We omit $\boldsymbol{\Phi}_{k-1}$ in (3.38) because we are assuming that all of the iteration vectors are used in each iteration. Additionally, we assume that the iteration vectors are partitioned into groups of equal size: $r_k^a = r_k^b = r_k^c \equiv r_k = \frac{q}{3}$ and that all turning vectors and turning-of-turning vectors are used in each iteration.

By construction, the iteration vectors stored in $\boldsymbol{X}_{k-1}$ are $\boldsymbol{M}$-orthonormal. Hence, the iteration vectors satisfy $\boldsymbol{X}_{k-1}^T \boldsymbol{M} \boldsymbol{X}_{k-1} = \boldsymbol{I}$. The analysis here follows the structure of analyses performed in [3] and [18] for the convergence of the basic subspace iteration method and the enriched subspace iteration method, respectively. To begin, let

$$\begin{aligned} \boldsymbol{X}_{k-1} = \boldsymbol{\Phi} \boldsymbol{\zeta}_{k-1}, \quad \text{where } \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_n \end{bmatrix} \\ \boldsymbol{\zeta}_{k-1} \in \mathbb{R}^{n \times q} \end{aligned} \tag{3.38}$$

The vectors stored in $\boldsymbol{\zeta}_{k-1}$ are the coordinates of the iteration vectors in the basis of the eigenvectors. The eigenvectors in $\boldsymbol{\Phi}$ are $\boldsymbol{M}$-orthonormal and $\boldsymbol{K}$-orthogonal and can be normalized such that the following equations are satisfied:

$$\begin{aligned} &\boldsymbol{\Phi}^T \boldsymbol{K} \boldsymbol{\Phi} = \boldsymbol{\Omega}^2 \quad \text{where } \boldsymbol{\Omega}^2 = \operatorname{diag}\left(\omega_1^2, \, \omega_2^2, \, \ldots, \, \omega_n^2\right) \\ &\boldsymbol{\Phi}^T \boldsymbol{M} \boldsymbol{\Phi} = \boldsymbol{I} \end{aligned} \tag{3.39}$$

Further, since $\boldsymbol{K}$ and $\boldsymbol{M}$ are symmetric and positive definite, it can be shown by

the spectral decomposition theorem that the generalized eigenvectors in $\boldsymbol{\Phi}$ are linearly independent. Therefore, $\boldsymbol{\Phi}$ is invertible and we obtain

$$\boldsymbol{M} = \boldsymbol{\Phi}^{-T}\boldsymbol{\Phi}^{-1} \quad \text{where } \boldsymbol{\Phi}^{-T} = \left(\boldsymbol{\Phi}^{-1}\right)^T = \left(\boldsymbol{\Phi}^T\right)^{-1} \tag{3.40}$$

Consequently, the system of equations in (3.38) can be solved uniquely for $\boldsymbol{\zeta}_{k-1}$:

$$\boldsymbol{\zeta}_{k-1} = \boldsymbol{\Phi}^{-1}\boldsymbol{X}_{k-1} \tag{3.41}$$

Further, by the $\boldsymbol{M}$-orthonormality of $\boldsymbol{X}_{k-1}$, it follows that

$$\begin{aligned} \boldsymbol{\zeta}_{k-1}^T\boldsymbol{\zeta}_{k-1} &= \left(\boldsymbol{\Phi}^{-1}\boldsymbol{X}_{k-1}\right)^T\boldsymbol{\Phi}^{-1}\boldsymbol{X}_{k-1} \\ &= \boldsymbol{X}_{k-1}^T\boldsymbol{\Phi}^{-T}\boldsymbol{\Phi}^{-1}\boldsymbol{X}_{k-1} \\ &= \boldsymbol{X}_{k-1}^T\boldsymbol{M}\boldsymbol{X}_{k-1} \\ &= \boldsymbol{I} \end{aligned} \tag{3.42}$$

Now, recall the first inverse iteration step of the E^2 method: we compute $\overline{\boldsymbol{X}}_k^a \in \mathbb{R}^{n\times r_k}$ by solving

$$\boldsymbol{K}\overline{\boldsymbol{X}}_k^a = \boldsymbol{M}\boldsymbol{X}_{k-1}^a \tag{3.43}$$

Now, let $\boldsymbol{\zeta}_{k-1}^a \in \mathbb{R}^{n\times r_k}$ denote the first r_k columns of $\boldsymbol{\zeta}_{k-1}$. Additionally, let $\overline{\boldsymbol{\zeta}}_k^a \in \mathbb{R}^{n\times r_k}$ denote the coordinates of $\overline{\boldsymbol{X}}_k^a$ in the basis of the eigenvectors. Hence,

$$\boldsymbol{X}_{k-1}^a = \boldsymbol{\Phi}\boldsymbol{\zeta}_{k-1}^a \quad \text{and} \quad \overline{\boldsymbol{X}}_k^a = \boldsymbol{\Phi}\overline{\boldsymbol{\zeta}}_k^a \tag{3.44}$$

Then, using this change of basis, the inverse iteration step of (3.43) can be written as

$$\begin{aligned} \boldsymbol{K}\overline{\boldsymbol{X}}_k^a &= \boldsymbol{M}\boldsymbol{X}_{k-1}^a \\ \boldsymbol{K}\boldsymbol{\Phi}\overline{\boldsymbol{\zeta}}_k^a &= \boldsymbol{M}\boldsymbol{\Phi}\boldsymbol{\zeta}_{k-1}^a \\ \boldsymbol{\Phi}^T\boldsymbol{K}\boldsymbol{\Phi}\overline{\boldsymbol{\zeta}}_k^a &= \boldsymbol{\Phi}^T\boldsymbol{M}\boldsymbol{\Phi}\boldsymbol{\zeta}_{k-1}^a \\ \boldsymbol{\Omega}^2\overline{\boldsymbol{\zeta}}_k^a &= \boldsymbol{\zeta}_{k-1}^a \end{aligned} \tag{3.45}$$

Since $\boldsymbol{K}$ and $\boldsymbol{M}$ are positive definite, $\boldsymbol{\Omega}^2$ is invertible, and the inverse is given by

$$\text{inv}\left(\boldsymbol{\Omega}^2\right) = \text{diag}\left(\frac{1}{\omega_1^2}, \frac{1}{\omega_2^2}, \ldots, \frac{1}{\omega_n^2}\right), \qquad \omega_i^2 > 0 \text{ for all } i = 1, 2, \ldots, n \tag{3.46}$$

Consequently, since the inverse of $\boldsymbol{\Omega}^2$ is well-defined, it follows from (3.45) that $\overline{\boldsymbol{\zeta}}_k^a$ is uniquely defined by

$$\overline{\boldsymbol{\zeta}}_k^a = \text{inv}\left(\boldsymbol{\Omega}^2\right) \boldsymbol{\zeta}_{k-1}^a \tag{3.47}$$

The quantity $\overline{\boldsymbol{\zeta}}_k^a$ represents $\overline{\boldsymbol{X}}_k^a$ in the eigenvector basis. It follows from (3.46) that, in the eigenvector basis, inverse iteration is performed by multiplying the current iteration vectors by $\text{inv}\left(\boldsymbol{\Omega}^2\right)$.

In the description of the algorithm in Section 3.1.1, we defined the quantities $\overline{\boldsymbol{Y}}_k \in \mathbb{R}^{n \times r_k}$ and $\overline{\boldsymbol{Z}}_k \in \mathbb{R}^{n \times r_k}$. Suppose that in each iteration, we use all of the turning vectors and turning-of-turning vectors to create forward turning and forward turning-of-turning vectors, respectively. In this case, $\overline{\boldsymbol{Y}}_k$ consists of all of the forward turning vectors and $\overline{\boldsymbol{Z}}_k$ consists of all of the forward turning-of-turning vectors. Then, the following relationships hold

$$\text{span}\left\{\overline{\boldsymbol{X}}_k^a, \overline{\boldsymbol{Y}}_k\right\} = \text{span}\left\{\overline{\boldsymbol{\zeta}}_k^a, \overline{\overline{\boldsymbol{\zeta}}}_k^a\right\}, \quad \text{where } \overline{\overline{\boldsymbol{\zeta}}}_k^a = \left[\text{inv}\left(\boldsymbol{\Omega}^2\right)\right]^2 \boldsymbol{\zeta}_{k-1}^a \tag{3.48}$$

$$\text{span}\left\{\overline{\boldsymbol{Y}}_k, \overline{\boldsymbol{Z}}_k\right\} = \text{span}\left\{\overline{\overline{\boldsymbol{\zeta}}}_k^a, \overline{\overline{\overline{\boldsymbol{\zeta}}}}_k^a\right\}, \quad \text{where } \overline{\overline{\overline{\boldsymbol{\zeta}}}}_k^a = \left[\text{inv}\left(\boldsymbol{\Omega}^2\right)\right]^3 \boldsymbol{\zeta}_{k-1}^a \tag{3.49}$$

Using the results from (3.48) and (3.49), we conclude that the current subspace is given in the eigenvector basis by

$$E_k = \text{span}\left\{\overline{\boldsymbol{\zeta}}_k^a, \overline{\overline{\boldsymbol{\zeta}}}_k^a, \overline{\overline{\overline{\boldsymbol{\zeta}}}}_k^a\right\} \tag{3.50}$$

Now, define $\boldsymbol{\Xi}_{k-1} \in \mathbb{R}^{n \times q}$ as follows:

$$\boldsymbol{\Xi}_{k-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \\ \xi_{q+1,1}^{(k-1)} & \xi_{q+1,2}^{(k-1)} & \cdots & \xi_{q+1,q}^{(k-1)} \\ \xi_{q+2,1}^{(k-1)} & \xi_{q+2,2}^{(k-1)} & \cdots & \xi_{q+2,q}^{(k-1)} \\ \vdots & \vdots & & \vdots \\ \xi_{n,1}^{(k-1)} & \xi_{n,2}^{(k-1)} & \cdots & \xi_{n,q}^{(k-1)} \end{bmatrix} \tag{3.51}$$

Let $\hat{\boldsymbol{e}}_1, \ldots, \hat{\boldsymbol{e}}_q$ denote the first q columns of the identity matrix. Assuming that the iteration vectors in $\boldsymbol{\zeta}_{k-1}$ are not deficient in $\hat{\boldsymbol{e}}_1, \ldots, \hat{\boldsymbol{e}}_q$, then the columns of $\boldsymbol{\zeta}_{k-1}$ can be linearly combined to obtain $\boldsymbol{\Xi}_{k-1}$. The first q rows of $\boldsymbol{\Xi}_{k-1}$ are the $q \times q$ identity matrix. Further, the column space of a matrix is not changed by forming linear combinations of the columns, so we conclude that $\text{span}\{\boldsymbol{\zeta}_{k-1}\} = \text{span}\{\boldsymbol{\Xi}_{k-1}\}$. Thus, $\boldsymbol{\zeta}_{k-1}$ and $\boldsymbol{\Xi}_{k-1}$ correspond to the same subspace. Now, the vectors $\boldsymbol{\zeta}_i^{a,(k-1)}$, for $i = 1, \ldots, r_k$ can be written as a linear combination of the corresponding i^{th} column of $\boldsymbol{\Xi}_{k-1}$ and a residual vector, given by $\boldsymbol{r}_i^{(k-1)}$:

$$\boldsymbol{\zeta}_i^{a,(k-1)} = \gamma_i^{(k-1)} \boldsymbol{\xi}_i^{(k-1)} + \epsilon_i^{(k-1)} \boldsymbol{r}_i^{(k-1)} \tag{3.52}$$

In (3.52), $\boldsymbol{\xi}_i^{(k-1)}$ denotes the i^{th} column of $\boldsymbol{\Xi}_{k-1}$, $\gamma_i^{(k-1)} \in \mathbb{R}$, $\epsilon_i^{(k-1)} \in \mathbb{R}$, and $\boldsymbol{r}_i^{(k-1)} \in \mathbb{R}^n$. As the subspace approaches convergence, the following limits are observed:

$$\lim_{k \to \infty} \boldsymbol{\xi}_i^{(k-1)} = \hat{\boldsymbol{e}}_i, \quad \lim_{k \to \infty} \gamma_i^{(k-1)} = 1, \quad \lim_{k \to \infty} \epsilon_i^{(k-1)} = 0 \tag{3.53}$$

We assume that the i^{th} iteration vector has reached its asymptotic convergence rate, which means that $\epsilon_i^{(k-1)} \ll 1$ and $\gamma_i^{(k-1)} = 1 + \mathcal{O}\left(\epsilon_i^{(k-1)}\right)$. Then, combining (3.52) with

the definition of $\overline{\overline{\overline{\boldsymbol{\zeta}}}}_k^a$ in (3.49), we have

$$\overline{\overline{\overline{\boldsymbol{\zeta}}}}_i^{a,(k)} = \gamma_i^{(k-1)} \left[\text{inv}\left(\boldsymbol{\Omega}^2\right)\right]^3 \boldsymbol{\xi}_i^{(k-1)} + \epsilon_i^{(k-1)} \left[\text{inv}\left(\boldsymbol{\Omega}^2\right)\right]^3 \boldsymbol{r}_i^{(k-1)} \tag{3.54}$$

In (3.54), $\overline{\overline{\overline{\boldsymbol{\zeta}}}}_i^{a,(k)}$ denotes the i^{th} column of $\overline{\overline{\overline{\boldsymbol{\zeta}}}}_k^a$. Then, it follows that,

$$\begin{aligned}
\frac{\left\|\left(\omega_i^2\right)^3 \overline{\overline{\overline{\boldsymbol{\zeta}}}}_i^{a,(k)} - \hat{\boldsymbol{e}}_i\right\|_2}{\left\|\boldsymbol{\zeta}_i^{a,(k-1)} - \hat{\boldsymbol{e}}_i\right\|_2} &= \frac{\left\|\gamma_i^{(k-1)} \left(\omega_i^2\right)^3 \left[\text{inv}\left(\boldsymbol{\Omega}^2\right)\right]^3 \boldsymbol{\xi}_i^{(k-1)} + \mathcal{O}\left(\epsilon_i^{(k-1)}\right) \boldsymbol{r}_i^{(k-1)} - \hat{\boldsymbol{e}}_i\right\|_2}{\left\|\gamma_i^{(k-1)} \boldsymbol{\xi}_i^{(k-1)} + \mathcal{O}\left(\epsilon_i^{(k-1)}\right) \boldsymbol{r}_i^{(k-1)} - \hat{\boldsymbol{e}}_i\right\|_2} \\
&= \frac{\left\|\left(\omega_i^2\right)^3 \left[\text{inv}\left(\boldsymbol{\Omega}^2\right)\right]^3 \boldsymbol{\xi}_i^{(k-1)} - \hat{\boldsymbol{e}}_i\right\|_2}{\left\|\boldsymbol{\xi}_i^{(k-1)} - \hat{\boldsymbol{e}}_i\right\|_2} + \mathcal{O}\left(\epsilon_i^{(k-1)}\right) \\
&= \sqrt{\frac{\sum_{j=q+1}^{n} \left(\frac{\omega_i^2}{\omega_j^2}\right)^6 \left(\xi_{j,i}^{(k-1)}\right)^2}{\sum_{j=q+1}^{n} \left(\xi_{j,i}^{(k-1)}\right)^2}} + \mathcal{O}\left(\epsilon_i^{(k-1)}\right) \\
&\leq \left(\frac{\omega_i^2}{\omega_{q+1}^2}\right)^3 + \mathcal{O}\left(\epsilon_i^{(k-1)}\right)
\end{aligned} \tag{3.55}$$

Finally, since $\boldsymbol{\zeta}_i^{a,(k)}$ is the best approximation of $\hat{\boldsymbol{e}}_i$ in the subspace E_k, it follows from the result of (3.55) that

$$\begin{aligned}
\frac{\left\|\boldsymbol{\zeta}_i^{a,(k)} - \hat{\boldsymbol{e}}_i\right\|_2}{\left\|\boldsymbol{\zeta}_i^{a,(k-1)} - \hat{\boldsymbol{e}}_i\right\|_2} &\leq \frac{\left\|\left(\omega_i^2\right)^3 \overline{\overline{\overline{\boldsymbol{\zeta}}}}_i^{a,(k)} - \hat{\boldsymbol{e}}_i\right\|_2}{\left\|\boldsymbol{\zeta}_i^{a,(k-1)} - \hat{\boldsymbol{e}}_i\right\|_2} \\
&\leq \left(\frac{\omega_i^2}{\omega_{q+1}^2}\right)^3 + \mathcal{O}\left(\epsilon_i^{(k-1)}\right), \qquad 1 \leq i \leq \frac{q}{3}
\end{aligned} \tag{3.56}$$

Therefore, assuming that the iteration vectors are ordered properly and are not $\boldsymbol{M}$-orthogonal to any of the target eigenvectors, we conclude from (3.56) that in the E^2 method, the first $q/3$ iteration vectors converge asymptotically proportional to $\left(\omega_i^2/\omega_{q+1}^2\right)^3$. Therefore, if $q = 3p$, the first p iteration vectors will converge asymptotically to the target subspace according to the rate $\left(\omega_i^2/\omega_{q+1}^2\right)^3$.

Further, since the eigenvalues are calculated using the Rayleigh quotient, which converges at a rate proportional to the square of the rate at which the eigenvectors converge, we conclude that the asymptotic convergence rate for eigenvalues using the E^2 method is $\left(\omega_i^2/\omega_{q+1}^2\right)^6$. For comparison purposes, the asymptotic convergence rates for the basic and enriched subspace iteration methods are provided in Table 3.5.

Let p_{k-1} denote the number of converged iteration vectors at the start of the k^{th} iteration. The important conclusion from this analysis is that for each iteration, no more than $\frac{q-p_{k-1}}{3}$ iteration vectors can converge at the asymptotic rate given in (3.56).

Method	Asymptotic Convergence Rate of Eigenvectors	Asymptotic Convergence Rate of Eigenvalues
Basic	$\omega_i^2/\omega_{q+1}^2$	$\left(\omega_i^2/\omega_{q+1}^2\right)^2$
Enriched	$\left(\omega_i^2/\omega_{q+1}^2\right)^2$	$\left(\omega_i^2/\omega_{q+1}^2\right)^4$
E^2	$\left(\omega_i^2/\omega_{q+1}^2\right)^3$	$\left(\omega_i^2/\omega_{q+1}^2\right)^6$

Table 3.5: Comparison of asymptotic convergence rates for the basic, enriched, and E^2 Bathe subspace iteration methods. For these results, we have assumed that all iteration vectors are used in each iteration and that all turning vectors and turning-of-turning vectors are used, as applicable, in each iteration. These convergence rates apply for $1 \leq i \leq q$ in the basic method, $1 \leq i \leq q/2$ in the enriched method and $1 \leq i \leq q/3$ in the E^2 method.

3.4 Test Problems

In this section, we provide sample solutions to illustrate the performance of the E^2 method on different test problems. We examine a variety of test problems, including:

- Small problems with $n < 4,000$ equations
- Large problems with $n > 100,000$ equations
- CPU time test problems with $n > 200,000$ equations

These tests were designed to assess the ability of the E^2 method to solve a wide variety of eigenvalue problems. The largest eigenvalue problem that is considered in this section is the finite element model of a 3D ring, which incorporates over 2 million degrees of freedom. In all of the test problems, no more than 400 of the least dominant eigenpairs are sought.

The new E^2 algorithm was written in Fortran 90. However, there are several subroutines that are called by the E^2 algorithm that were written in Fortran 77. Specifically, Fortran 77 routines were used to multiply the iteration vectors by $\boldsymbol{M}$, to perform the Cholesky decomposition of $\boldsymbol{K}$, to perform the inverse iteration step, to solve the projected eigenvalue problem using the Jacobi method, and to perform the Sturm sequence check after reaching convergence. These subroutines are available in reference [5].

The following test problems were obtained from finite element systems that modeled two-dimensional and three-dimensional structures. The finite element models were created using a partial differential equations package that is an add-on for the computer software MATLAB. The stiffness and mass matrices that were generated in MATLAB were converted to a skyline representation, which is a sparse storage scheme that is described in reference [5].

For some of the test problems, random starting vectors were used. For other problems, the Bathe starting vectors were used, which are described in reference [5]. The tables used to describe each example problem are explicit with regard to the method used to construct the starting vectors.

All test problems were conducted on a Mac Pro desktop computer using a single core with a 2.71 GHz Intel Core i5 processor and 32 GB of random access memory. For these results, we did not use a parallel implementation of the algorithm.

3.4.1 Small Test Problems

In this section, we consider the results of two small test problems. These problems examine the dynamics of two different two-dimensional vibrating membranes. These small test problems were used to assist in debugging the algorithm during its early stages of development. Additionally, these problems were used to test the theoretical asymptotic convergence rate of the E^2 method that was derived in Section 3.3. For these small problems, we present the asymptotic convergence rates that are observed for the basic, enriched, and E^2 subspace iteration methods.

Test Problem 1 - Vibrating Membrane on Rectangular Domain

Number of Equations	3,876
Half-Bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$	106
Order of FE Basis Functions	Linear
Number of Eigenvalues Sought	400
Number of Iteration Vectors Used	800
Convergence Tolerance	$< 10^{-8}$
Starting Vector Type	Bathe

Table 3.6: Test problem 1 details.

In this test problem, the generalized eigenvalues of a vibrating square membrane are calculated. The stiffness and mass matrices for this problem were determined from the discretization of the wave partial differential equation using the finite element method. An adaptive triangular mesh was used to model the problem geometry. The size of the domain was 1m $\times$ 1m. For this problem, the boundary conditions were fixed (homogeneous, Dirichlet) on all sides.

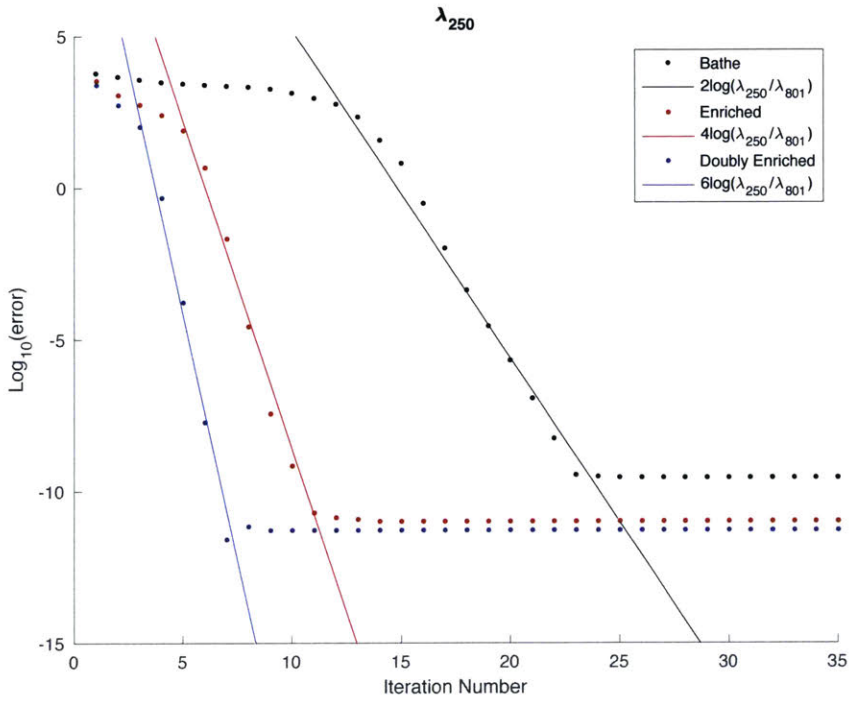

Figure 3-7: Convergence results for the 250^{th} smallest eigenvalue of the membrane problem. The asymptotic convergence rates for the basic, enriched, and E^2 methods are depicted.

As demonstrated in Figure 3-7, the observed asymptotic convergence rates for the basic, enriched, and E^2 methods are consistent with their theoretical rates that are reported in Table 3.5. It is clear from Figure 3-7 that, at least for the 250^{th} smallest eigenvalue, the E^2 algorithm converged in fewer iterations than the enriched method. This is important because the computational savings of the E^2 method are observed when the E^2 algorithm converges in fewer iterations than the other methods.

For a given problem, if the E^2 method converges in approximately the same number of iterations as the enriched method, then using the E^2 method may not provide any computational savings. This is because each iteration of the E^2 method usually requires more CPU time to complete than each iteration of the enriched method since the E^2 method requires the additional computation of the turning-of-turning vectors. Our experience suggests that even when the enriched method out-performs the E^2 method, the E^2 method is not much slower than the enriched method.

Test Problem 2 - Vibrating Membrane on Rectangular Domain with Square Hole

Number of Equations	3,777
Half-Bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$	2,905
Order of FE Basis Functions	Linear
Number of Eigenvalues Sought	100
Number of Iteration Vectors Used	200
Convergence Tolerance	$< 10^{-8}$
Starting Vector Type	Bathe

Table 3.7: Test problem 2 details.

The eigenvalues of a vibrating square membrane are calculated for a more complicated geometry than the simple square geometry considered in the previous example. The stiffness and mass matrices for this problem were determined from the discretization of the wave partial differential equation using the finite element method. An adaptive triangular mesh was used to model the problem geometry. The size of the domain was 1m $\times$ 1m with a square hole of size 1/3m $\times$ 1/3m removed from the center. For this problem, the boundary conditions were fixed (homogeneous, Dirichlet) on all sides.

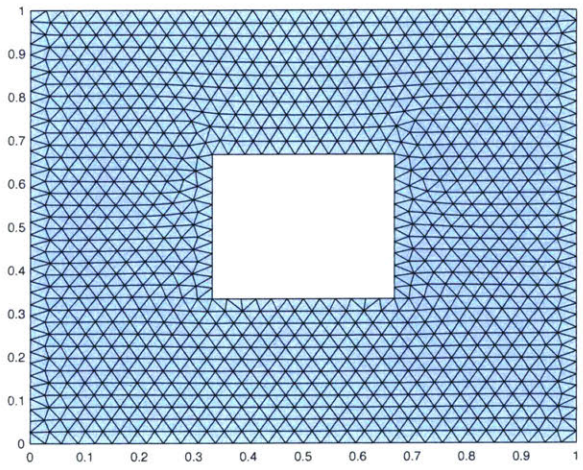

Figure 3-8: Geometry and mesh of the membrane with square hole problem.

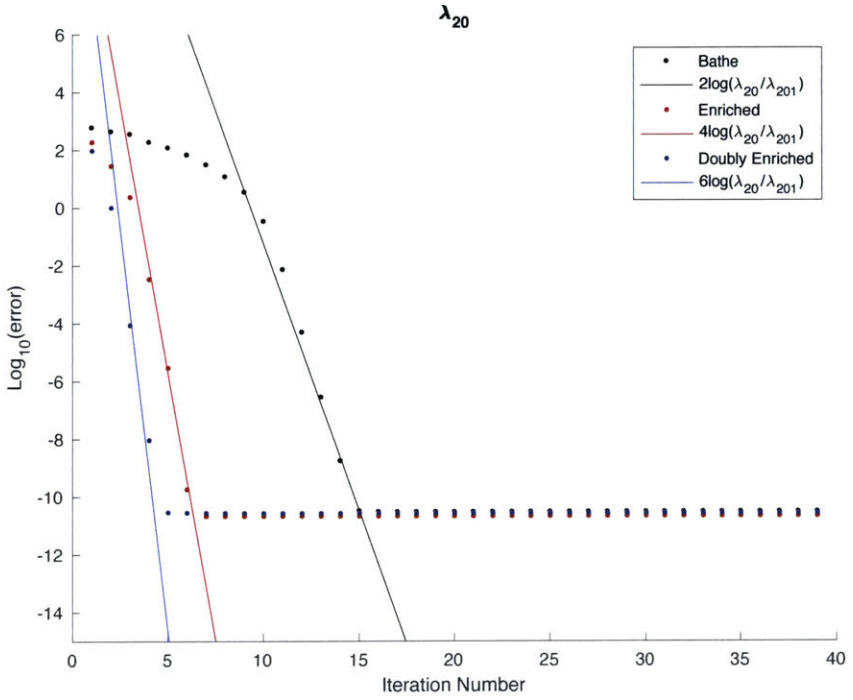

Figure 3-9: Convergence results for the 20^{th} smallest eigenvalue of the membrane with square hole problem. The asymptotic convergence rates for the basic, enriched, and E^2 methods are depicted.

As demonstrated in Figure 3-9, the asymptotic convergence rates for the basic, enriched, and E^2 methods are consistent with their theoretical rates that are reported in Table 3.5. In this case, the E^2 algorithm converged in two less iterations than the enriched method. However, both the enriched and E^2 methods converged in many less iterations than the basic method.

3.4.2 Large Test Problems

Having now examined the performance of the E^2 method on two small test problems, we turn our attention to larger problems on the scale of problems that are commonly seen in actual engineering analyses. The purpose of these large problems is to examine the scalability of the E^2 method.

For each of the four large test problems, the following information is supplied:

- A table showing the details of the problem, including the number of equations and

the half-bandwidth of the stiffness and mass matrices.

- A figure depicting the problem geometry.

- Observed convergence rates of the enriched and E^2 methods for a single representative eigenvalue.

- A figure showing the different convergence zones for the E^2 method. The convergence zones are identified as follows:

  - **Pre-asymptotic convergence zone:** this occurs in the first few iterations of the E^2 method when the initial subspaces are not close enough to the target subspace for the asymptotic convergence rate to be observed.

  - **Asymptotic convergence zone:** this occurs when the iteration vectors converge according to the rate $\left(\omega_i^2/\omega_{q+1}^2\right)^3$, as predicted in Section 3.3. In this zone, there are plenty of non-converged iteration vectors, so turning vectors and turning-of-turning vectors can be used to accelerate convergence to the target subspace.

  - **Post-asymptotic convergence zone:** this occurs when the subspace has almost converged to the target subspace and the iteration vectors are no longer turning significantly enough to be used as turning vectors or turning-of-turning vectors. If the subspace is still turning enough to compute turning vectors, then the E^2 method will converge at the rate of the enriched method, otherwise, if no turning vectors are calculated, then the E^2 method will converge at the rate of the basic method.

  - **Converged zone:** this occurs when the iteration vector has converged and does not continue iterating.

- A table showing the CPU time used in each iteration of the E^2 method. The numbers of turning vectors and turning-of-turning vectors used in each iteration of the E^2 method are also provided.

A summary of the results from the following four large test problems is given below in Table 3.8.

Problem number	Method	Number of iterations to reach convergence	Number of equations	Bandwidth	Number of eigenpairs sought	Number of iteration vectors vectors used	Total CPU time
3	Enriched	20	112,758	2,647	100	140	2,942
3	E^2	17	112,758	2,647	100	140	2,643
4	Enriched	49	525,060	342	100	160	1,968
4	E^2	36	525,060	342	100	160	1,571
5	Enriched	19	1,004,262	411	100	140	4,520
5	E^2	19	1,004,262	411	100	140	4,651
6	Enriched	21	2,013,571	756	100	140	15,729
6	E^2	19	2,013,571	756	100	140	15,302

Table 3.8: Summary of the performances of the enriched and E^2 methods applied to test problems 3-6. In terms of CPU time required to reach convergence, the enriched method out-performed the E^2 method in test problem 5. In test problem 6, the results between the two methods were similar, but the E^2 method converged in about 2% less time than the enriched method in this problem. For test problems 3 and 4, the E^2 method provided computational savings of roughly 10% and 20%, respectively, compared to the enriched method.

Test Problem 3 - 3D Bracket with Two Holes

In this test problem, we examine a finite element model of a 3D bracket with two holes, as shown in Figure 3-10. Convergence results for the enriched and E^2 method are presented in Figures 3-12 and 3-13 for the 70^{th} smallest eigenvalue.

Number of Equations	112,758
Half-Bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$	2,647
Order of FE Basis Functions	Quadratic
Number of Eigenvalues Sought	100
Number of Iteration Vectors Used	140
Convergence Tolerance	$< 10^{-6}$
Starting Vector Type	Bathe

Table 3.9: Test problem 3 details.

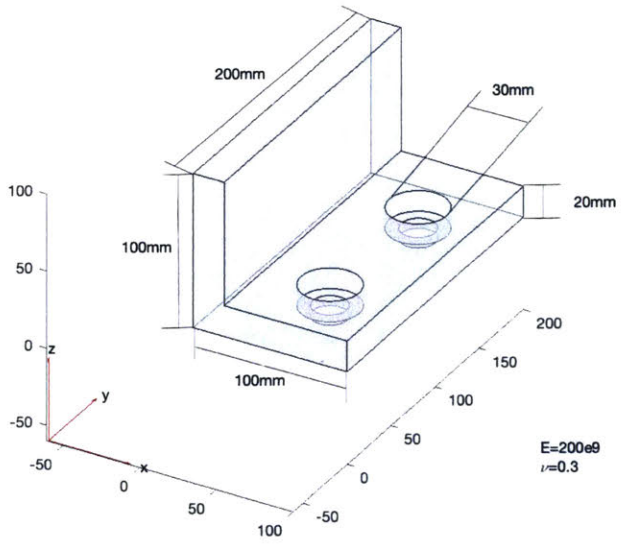

Figure 3-10: Geometry of the 3D bracket with two holes problem.

In each convergence zone, the E^2 method converges at a different rate. In the post-asymptotic convergence zone, the E^2 method converges more slowly than in the asymptotic convergence zone due to the E^2 method using less turning vectors and turning-of-turning vectors as the iteration vectors become very close to the target subspace.

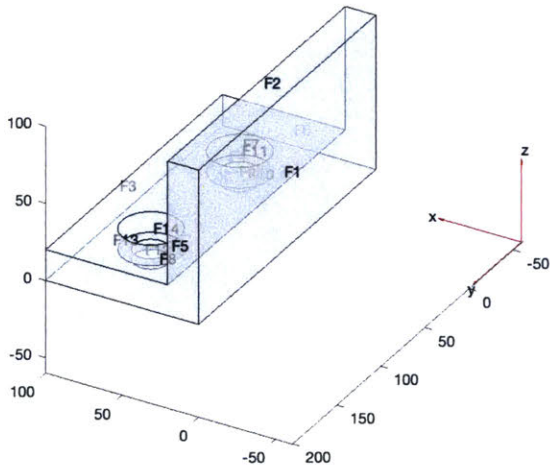

Figure 3-11: The structure was fixed at the surface denoted F1.

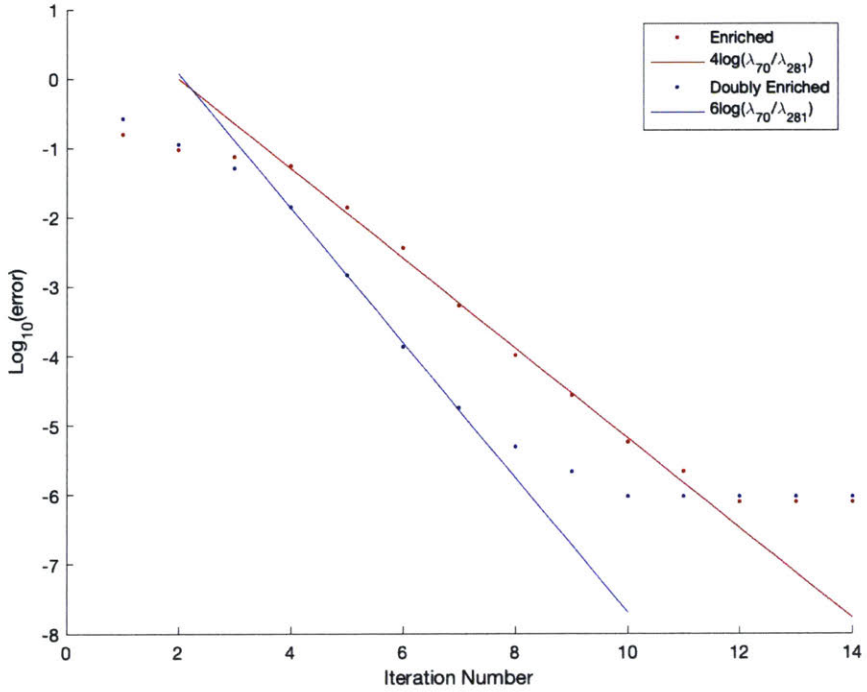

Figure 3-12: Convergence results for the 3D bracket with two holes problem. Asymptotic convergence rates for the enriched and E^2 methods are depicted for the 70^{th} smallest eigenvalue.

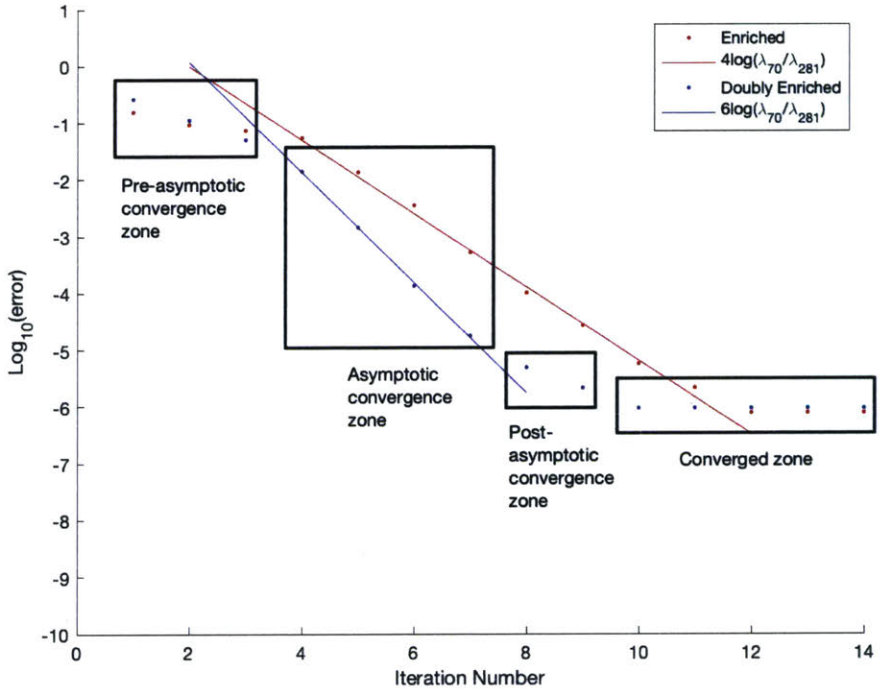

Figure 3-13: Convergence results for the 3D bracket with two holes problem with the different convergence zones for the E^2 method indicated.

E^2 Bathe Subspace Iteration Method				
Iteration number	Number of turning vectors used in iteration	Number of turning-of-turning vectors used in iteration	Cumulative number of converged vectors after iteration	CPU time (sec) used in iteration/ Rounded cumulative CPU time (sec)
1	46	42	0	309/309
2	46	28	5	257/566
3	45	16	14	250/816
4	27	3	23	228/1044
5	35	1	28	211/1255
6	31	2	42	195/1450
7	25	5	55	167/1617
8	21	2	65	150/1767
9	19	1	65	136/1903
10	15	0	72	132/2035
11	13	0	78	123/2158
12	16	0	81	109/2267
13	14	1	88	103/2370
14	14	1	92	95/2465
15	11	2	94	88/2553
16	9	0	96	84/2637
17	8	0	100	6/2637

Table 3.10: CPU time used in each iteration of the E^2 subspace iteration method for calculating the $p = 100$ smallest eigenvalues of the 3D bracket with two holes problem.

Test Problem 4 - 3D Wall

In this test problem, we examine a finite element model of a 3D wall, as shown in Figure 3-14. Convergence results for the enriched and E^2 method are presented in Figures 3-16 and 3-17 for the 40^{th} smallest eigenvalue.

Number of Equations	525,060
Half-Bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$	342
Order of FE Basis Functions	Linear
Number of Eigenvalues Sought	100
Number of Iteration Vectors Used	160
Convergence Tolerance	$< 10^{-6}$
Starting Vector Type	Bathe

Table 3.11: Test problem 4 details.

In each convergence zone, the E^2 method converges at a different rate. In the post-asymptotic convergence zone, the E^2 method converges more slowly than in the asymptotic convergence zone due to the E^2 method using less turning vectors and turning-of-turning vectors as the iteration vectors become very close to the target subspace.

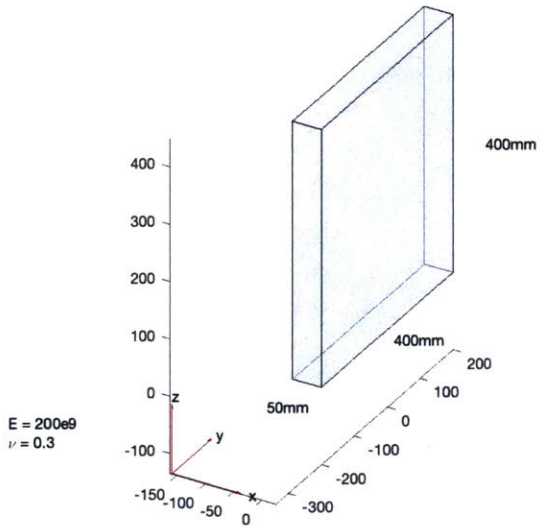

Figure 3-14: Geometry of the 3D wall problem.

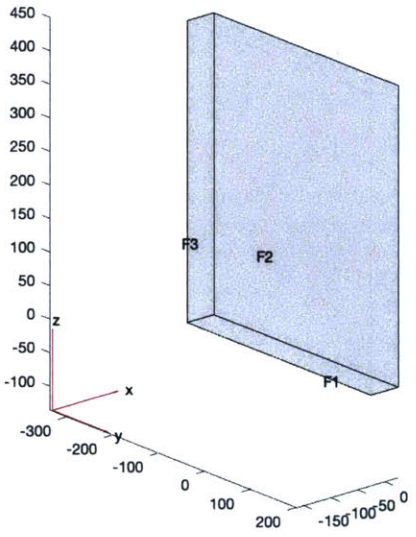

Figure 3-15: The structure was fixed at the surface denoted F1.

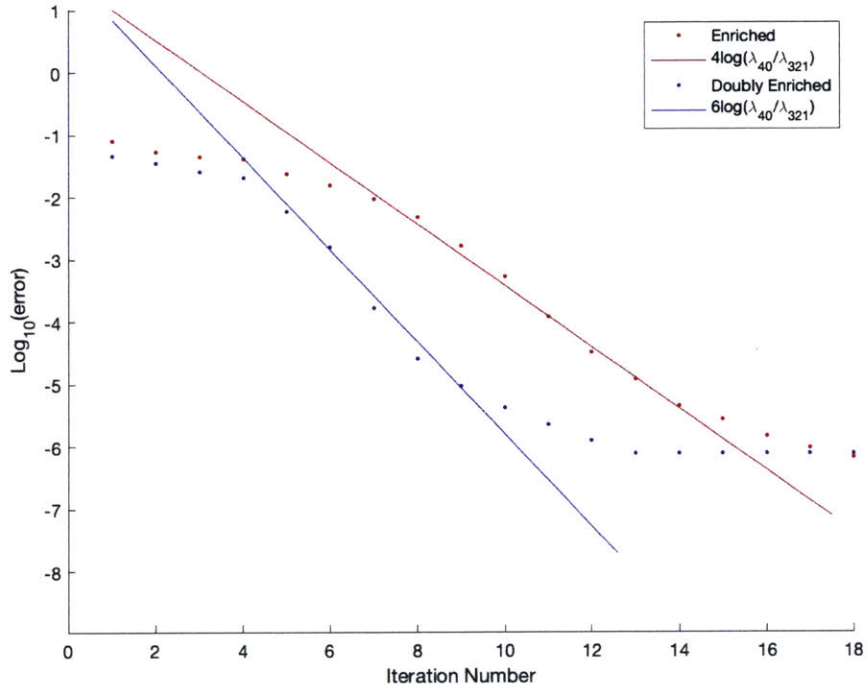

Figure 3-16: Convergence results for the 3D wall problem. Asymptotic convergence rates for the enriched and E^2 methods are depicted for the 40^{th} smallest eigenvalue.

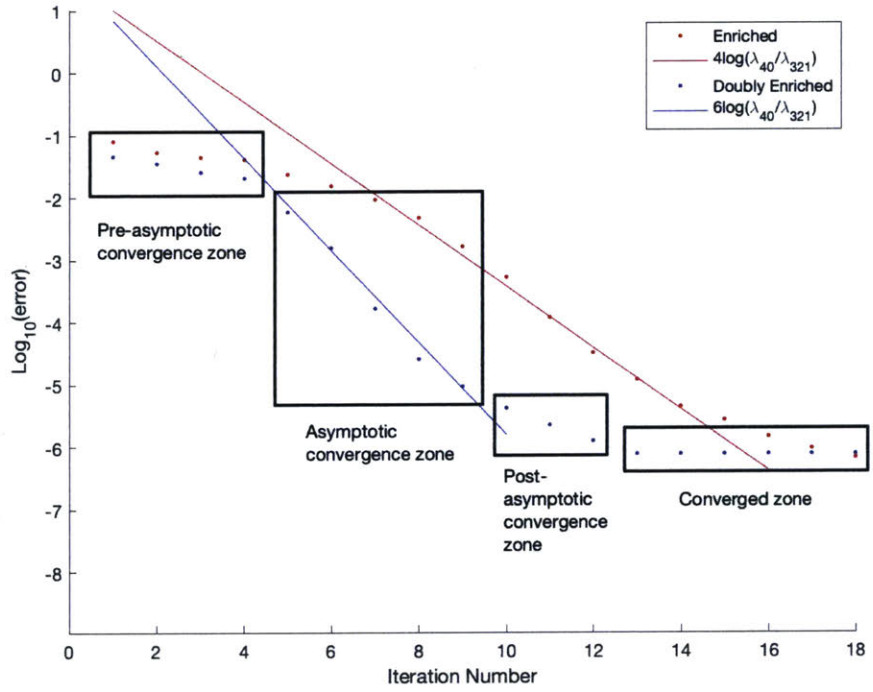

Figure 3-17: Convergence results for the 3D wall problem with the different convergence zones for the E^2 method indicated.

E^2 Bathe Subspace Iteration Method				
Iteration number	Number of turning vectors used in iteration	Number of turning-of-turning vectors used in iteration	Cumulative number of converged vectors after iteration	CPU time (sec) used in iteration/ Rounded cumulative CPU time (sec)
1	53	53	0	71/71
2	53	39	0	69/140
3	53	34	2	69/209
4	47	25	7	68/277
5	40	21	11	64/341
6	40	15	13	60/401
7	33	11	22	58/459
8	33	9	29	57/516
9	31	10	29	54/570
10	26	8	36	52/622
11	25	8	39	50/672
12	24	8	39	49/721
13	22	10	40	47/768
14	24	10	48	46/814
15	21	9	48	45/859
16	19	10	48	43/902
17	17	8	51	42/944
18	18	6	51	41/985
19	16	4	54	40/1025
20	16	2	54	39/1064
21	15	1	55	38/1102
22	17	1	55	37/1139
23	14	1	55	37/1176
24	16	1	55	36/1212
25	15	1	56	35/1247
26	18	1	60	34/1281
27	17	0	60	33/1314
28	16	0	60	33/1347
29	17	1	69	32/1379
30	18	2	69	31/1410
31	17	3	99	30/1440
32	18	5	99	29/1469
33	16	4	99	29/1498
34	14	3	99	28/1526
35	13	2	99	28/1554
36	13	0	100	17/1571

Table 3.12: CPU time used in each iteration of the E^2 subspace iteration method for calculating the $p = 100$ smallest eigenvalues of the 3D wall problem.

Test Problem 5 - 3D Pipe

In this test problem, we examine a finite element model of a 3D pipe, as shown in Figure 3-18. Convergence results for the enriched and E^2 method are presented in Figures 3-20 and 3-21 for the 60^{th} smallest eigenvalue.

Number of Equations	1,004,262
Half-Bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$	411
Order of FE Basis Functions	Linear
Number of Eigenvalues Sought	100
Number of Iteration Vectors Used	140
Convergence Tolerance	$< 10^{-6}$
Starting Vector Type	Random

Table 3.13: Test problem 5 details.

In this problem, random starting vectors were used to show that the E^2 method can obtain the theoretical convergence rate without using the Bathe starting vectors.

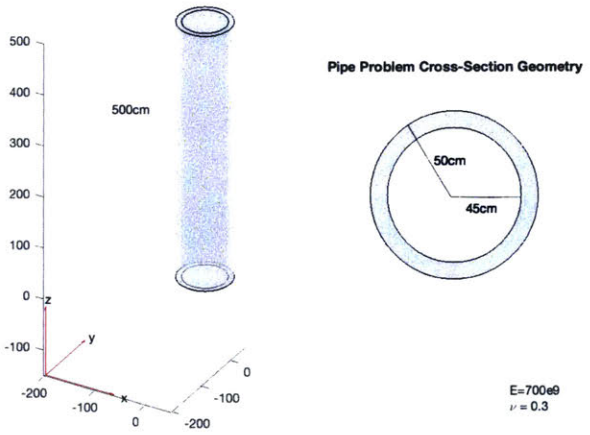

Figure 3-18: Geometry of the 3D pipe problem.

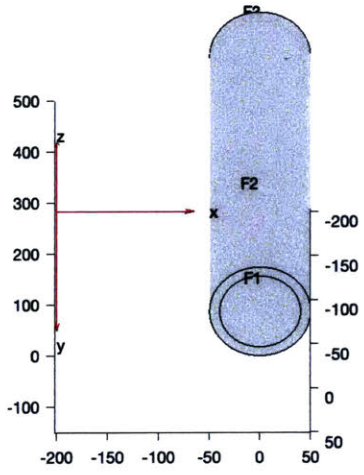

Figure 3-19: The structure was fixed at the surface denoted F1.

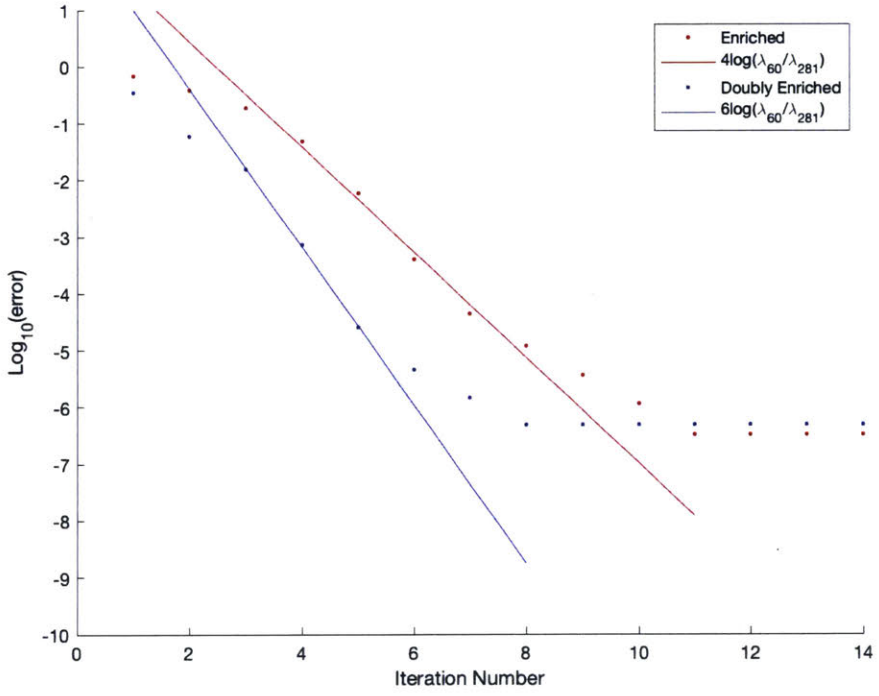

Figure 3-20: Convergence results for the 3D pipe problem. Asymptotic convergence rates for the enriched and E^2 methods are depicted for the 60^{th} smallest eigenvalue.

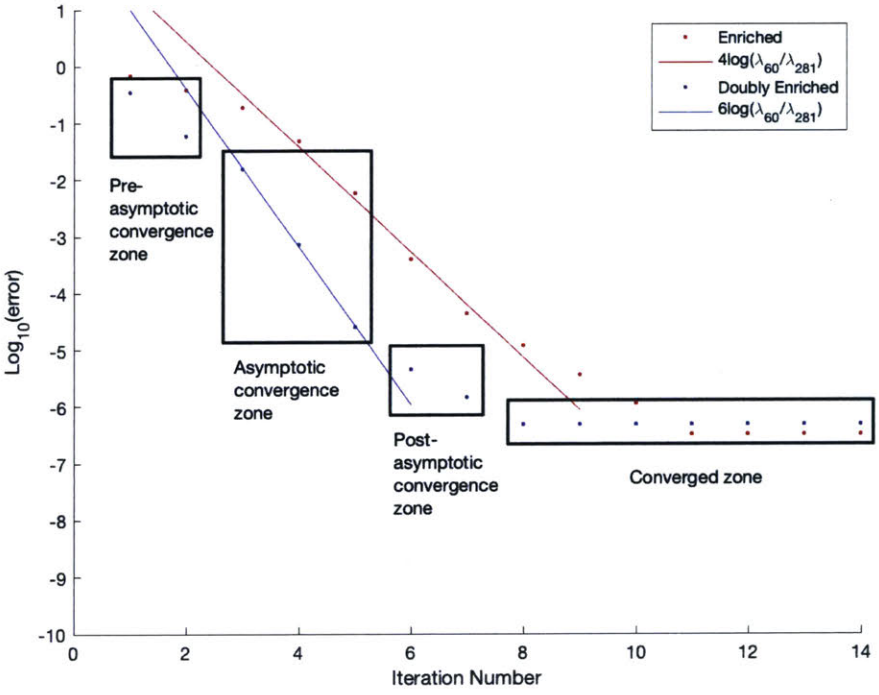

Figure 3-21: Convergence results for the 3D pipe problem with the different convergence zones for the E^2 method indicated.

E^2 Bathe Subspace Iteration Method				
Iteration number	Number of turning vectors used in iteration	Number of turning-of-turning vectors used in iteration	Cumulative number of converged vectors after iteration	CPU time (sec) used in iteration/ Rounded cumulative CPU time (sec)
1	46	44	1	411/411
2	46	33	4	406/817
3	45	9	10	391/1208
4	41	6	23	371/1579
5	39	5	31	338/1917
6	32	5	46	302/2219
7	26	2	52	271/2490
8	20	0	60	254/2744
9	19	0	65	235/2979
10	17	0	67	222/3201
11	17	0	79	209/3410
12	16	1	81	185/3595
13	13	1	88	174/3769
14	11	0	90	157/3926
15	9	0	90	154/4080
16	7	0	92	151/4231
17	6	0	94	146/4377
18	7	0	97	141/4518
19	7	0	100	133/4651

Table 3.14: CPU time used in each iteration of the E^2 subspace iteration method for calculating the $p = 100$ smallest eigenvalues of the 3D pipe problem.

Test Problem 6 - 3D Ring

In this test problem, we examine a finite element model of a 3D ring, as shown in Figure 3-22. Convergence results for the enriched and E^2 method are presented in Figures 3-23 and 3-24 for the 60^{th} smallest eigenvalue.

In Figure 3-24, we have indicated different convergence zones for the E^2 method. In each convergence zone, the E^2 method converges at a different rate. We note that in the post-asymptotic convergence zone, the E^2 method converges more slowly than in the asymptotic convergence zone. This is due to the E^2 method using less turning vectors and turning-of-turning vectors as the iteration vectors become very close to the target subspace. The number of turning vectors and turning-of-turning vectors used in each iteration of the E^2 method for this problem are reported in Table 3.16.

In this problem, random starting vectors were used to show that the E^2 method can obtain the theoretical convergence rate without using the Bathe starting vectors.

Number of Equations	2,013,571
Half-Bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$	756
Order of FE Basis Functions	Linear
Number of Eigenvalues Sought	100
Number of Iteration Vectors Used	140
Convergence Tolerance	$< 10^{-6}$
Starting Vector Type	Random

Table 3.15: Test problem 6 details.

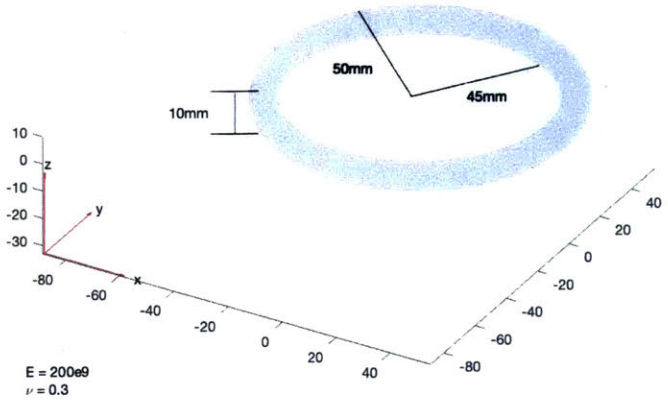

Figure 3-22: Geometry of the 3D ring problem. The structure was fixed at the bottom of the ring.

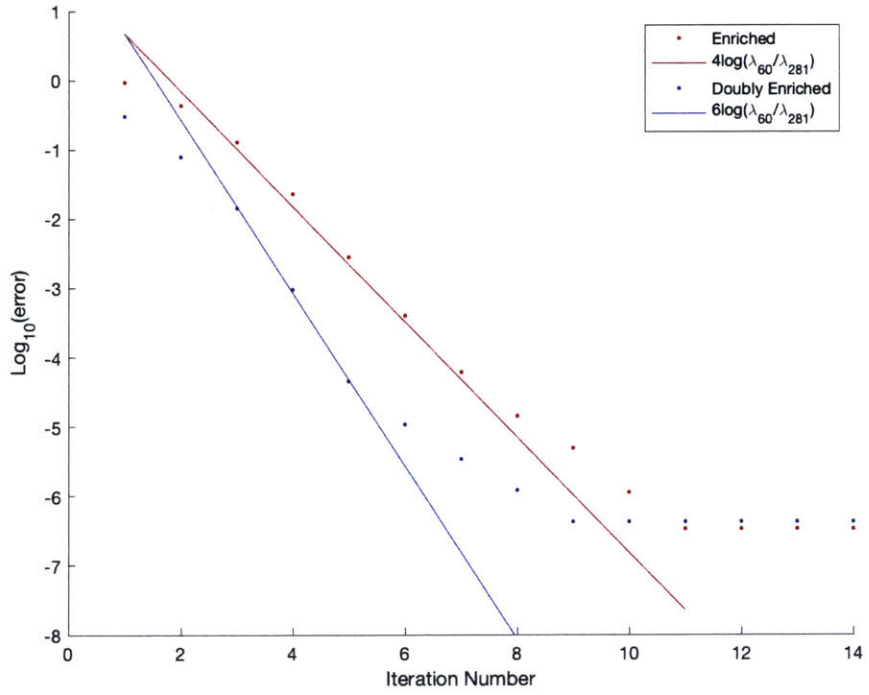

Figure 3-23: Convergence results for the 3D ring problem. Asymptotic convergence rates for the enriched and E^2 methods are depicted for the 60^{th} smallest eigenvalue.

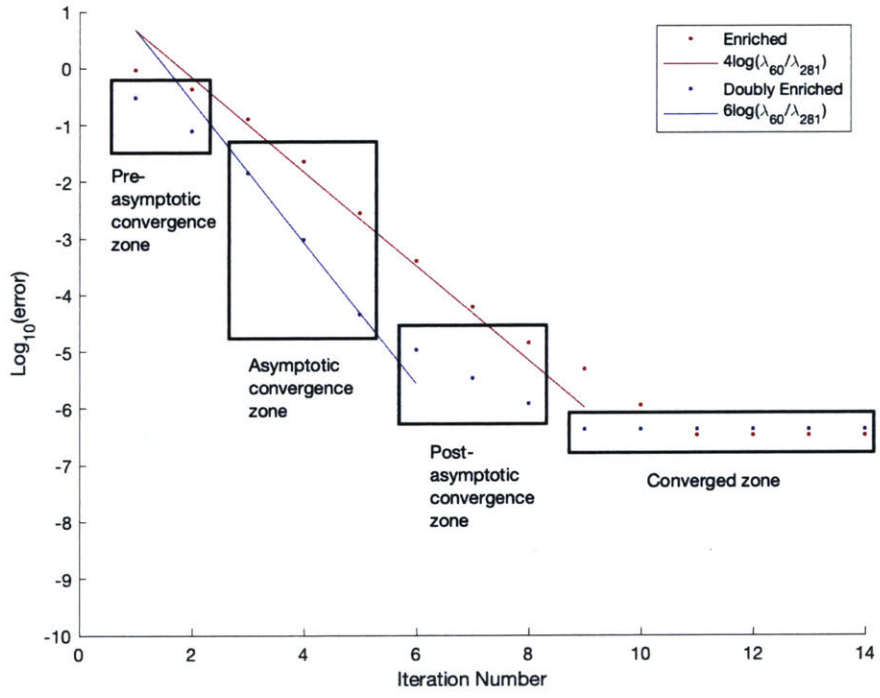

Figure 3-24: Convergence results for the 3D ring problem with the different convergence zones for the E^2 method indicated.

E^2 Bathe Subspace Iteration Method				
Iteration number	Number of turning vectors used in iteration	Number of turning-of-turning vectors used in iteration	Cumulative number of converged vectors after iteration	CPU time (sec) used in iteration/ Rounded cumulative CPU time (sec)
1	46	44	1	1367/1367
2	46	32	3	1345/2712
3	45	7	9	1316/4028
4	38	4	23	1255/5283
5	37	5	33	1125/6408
6	31	4	46	1013/7421
7	25	2	50	899/8320
8	19	0	59	868/9188
9	18	0	61	786/9974
10	15	0	71	730/10704
11	15	0	78	667/11371
12	17	0	81	595/11966
13	13	1	87	548/12514
14	11	0	87	521/13035
15	7	0	92	500/13535
16	7	0	93	483/ 14018
17	7	0	99	447/14465
18	5	0	99	418/14883
19	5	0	100	419/15302

Table 3.16: CPU time used in each iteration of the E^2 subspace iteration method for calculating the $p = 100$ smallest eigenvalues of the 3D ring problem.

3.4.3 CPU Time Test Problems

In test problems 1-6, we observed that the E^2 method converges according to the theoretical rate derived in Section 3.3. In this section, we determine if this accelerated convergence rate actually translates to computational savings with respect to the CPU time required to reach convergence. This is important to examine because, in general, each iteration of the E^2 method is more computationally expensive than each iteration of the enriched method due to the additional cost of computing the turning-of-turning vectors. Therefore, we need to determine if the extra cost of each iteration of the E^2 method is worth the benefit of the accelerated asymptotic convergence rate that the E^2 method provides. Additionally, in this section we examine the effect of the dimension of the subspace, q, on the CPU time required to reach convergence.

For these CPU time test problems, we consider the solution of eigenvalue problems

arising from the finite element analysis of three-dimensional structures using at least 200,000 degrees of freedom. Each test problem is solved using both the enriched and the E^2 method. The CPU time results for each method are compared in order to assess the potential computational savings of the E^2 method. For each test problem, we report the "percent savings" that would be obtained using the E^2 method instead of the enriched method. The percent savings results were calculated as follows:

$$\text{percent savings} = \frac{\text{CPU time for enriched method} - \text{CPU time for } E^2 \text{ method}}{\text{CPU time for enriched method}} \tag{3.57}$$

For some of the test problems, the percent savings measure is found to be negative, which indicates that for that particular test problem, the enriched method out-performed the E^2 method. When this happened, it was usually when q was close to $2p$. The E^2 method was usually faster than the enriched method when q was selected in the interval $p < q < \frac{3p}{2}$. We have highlighted the instances of negative percent savings in red text for quick identification.

We note that the solution times reported in the following tables do not reflect the maximum performance of the E^2 algorithm. This is because we have not taken full advantage of the sparsity of the stiffness and mass matrices. Specifically, using our current storage scheme, all of the zeros of each column that occur between the main diagonal and the last non-zero entry of the column are stored and treated as non-zeros. Further, our matrix multiplication routine and also the reduction and back substitution routines are not optimized to take advantage of the sparsity within the bandwidths of $\boldsymbol{K}$ and $\boldsymbol{M}$. Consequently, for the following results, many multiplications of zeros are unnecessarily performed, which leads to longer solution times. Using more efficient routines, it would be possible to reduce many unnecessary computations. However, since the same routines are used for the enriched and E^2 methods, the results in this section are comparable for our purposes.

Finally, we observe that the CPU time test results suggest that the solution time using the E^2 method scales linearly with respect to the number of eigenpairs sought.

Test Problem 7 - 3D Bracket with Connector and Holes

Number of Equations	205,929
Half-Bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$	2,612
Order of FE Basis Functions	Quadratic
Number of Eigenvalues Sought	100, 200, 400
Number of Iteration Vectors Used	Various
Convergence Tolerance	$< 10^{-6}$
Starting Vector Type	Bathe

Table 3.17: Test problem 7 details.

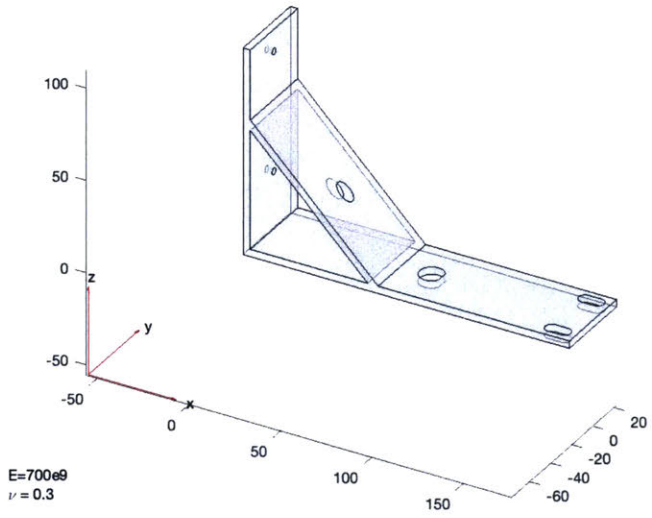

Figure 3-25: Geometry of the 3D bracket with connector and holes problem.

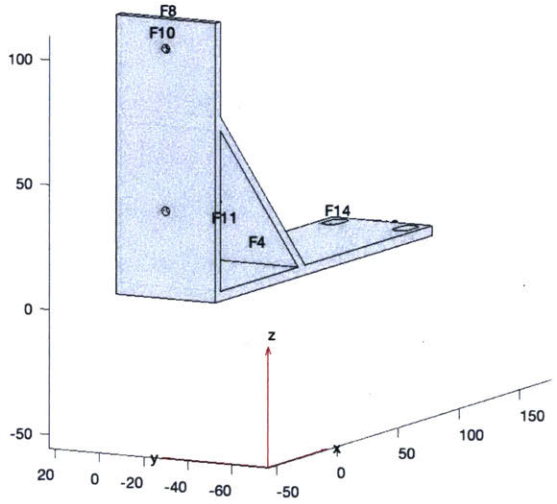

Figure 3-26: The structure was fixed at the surface denoted F10.

Number of Eigenvalues Sought	Number of Iteration Vectors	CPU Time (sec) for E-Subspace Iteration	CPU Time (sec) for E^2-Subspace Iteration	Percent Savings
100	110	4,324	3,899	9.82%
100	140	3,319	3,193	3.79%
100	170	3,230	3,424	-6.00%
100	200	3,246	3,284	-1.17%
200	220	8,329	7,871	5.49%
200	280	7,217	6,749	6.48%
200	340	7,245	7,326	-1.11%
200	400	7,569	7,650	-1.07%
400	440	20,580	20,221	1.74%
400	560	17,433	16,863	3.26%
400	680	17,336	18,727	-8.02%
400	800	17,212	17,662	-2.61%

Table 3.18: CPU time results for test problem 7.

Test Problem 8 - 3D Heat Sink

Number of Equations	303,204
Half-Bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$	1,985
Order of FE Basis Functions	Quadratic
Number of Eigenvalues Sought	100, 200, 400
Number of Iteration Vectors Used	Various
Convergence Tolerance	$< 10^{-6}$
Starting Vector Type	Bathe

Table 3.19: Test problem 8 details.

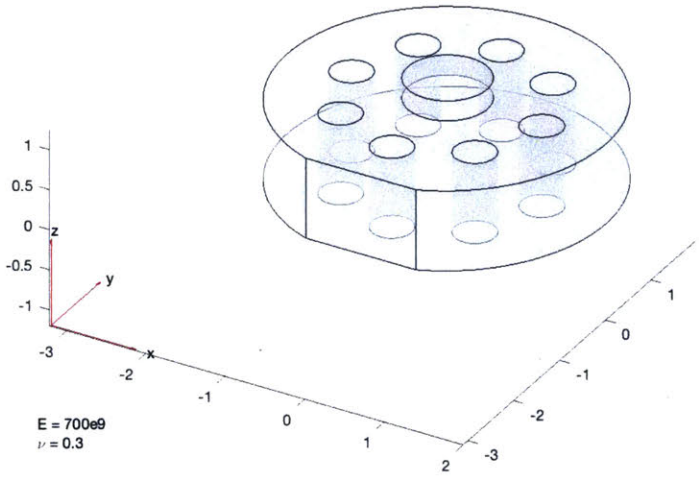

Figure 3-27: Geometry of the 3D heat sink problem.

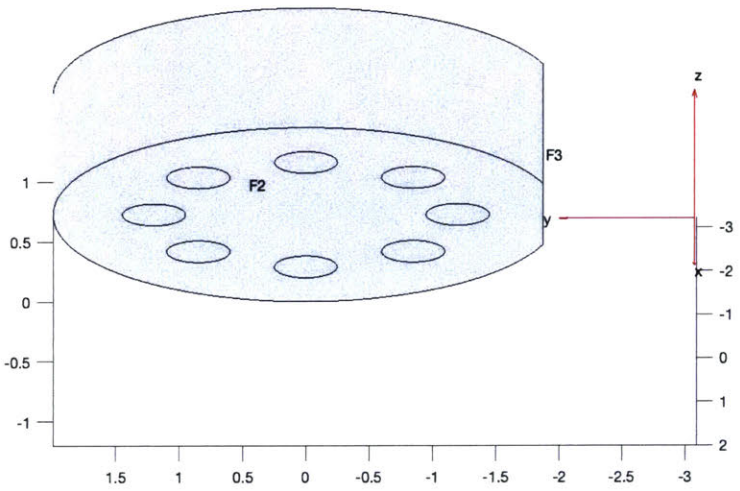

Figure 3-28: The structure was fixed at the surface denoted F2.

Number of Eigenvalues Sought	Number of Iteration Vectors	CPU Time (sec) for E-Subspace Iteration	CPU Time (sec) for E^2-Subspace Iteration	Percent Savings
100	110	41,177	31,519	23.45%
100	140	25,675	23,290	9.28%
100	170	23,324	22,426	3.85%
100	200	24,361	21,823	10.41%
200	220	61,239	54,308	11.31%
200	280	45,645	42,515	6.85%
200	340	44,136	41,921	5.01%
200	400	44,768	43,514	2.80%
400	440	112,860	101,278	10.26%
400	560	83,632	77,841	6.92%
400	680	82,391	79,275	3.78%
400	800	87,198	81,802	6.18%

Table 3.20: CPU time results for test problem 8.

Test Problem 9 - 3D Bracket with One Hole

Number of Equations	420,846
Half-Bandwidth of $\boldsymbol{K}$ and $\boldsymbol{M}$	8,975
Order of FE Basis Functions	Quadratic
Number of Eigenvalues Sought	100, 200, 400
Number of Iteration Vectors Used	Various
Convergence Tolerance	$< 10^{-6}$
Starting Vector Type	Bathe

Table 3.21: Test problem 9 details.

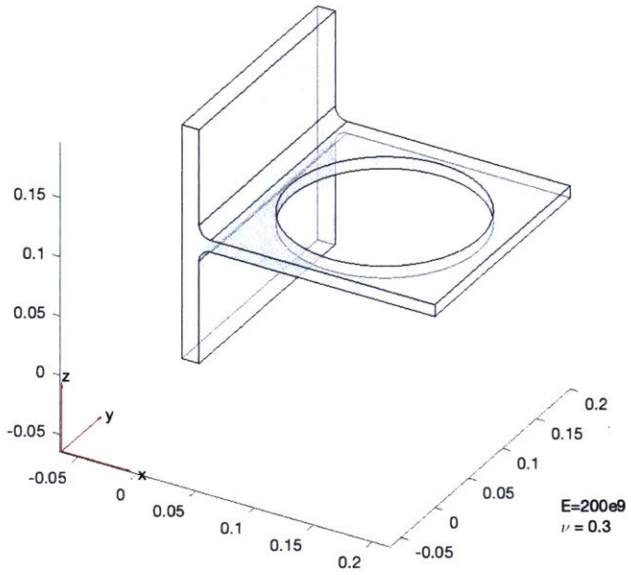

Figure 3-29: Geometry of the 3D bracket with one hole problem.

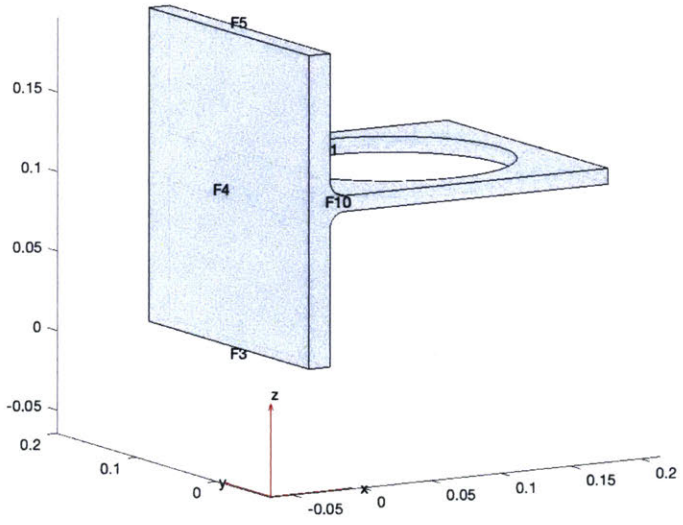

Figure 3-30: The structure was fixed at the surface denoted F4.

Number of Eigenvalues Sought	Number of Iteration Vectors	CPU Time (sec) for E-Subspace Iteration	CPU Time (sec) for E^2-Subspace Iteration	Percent Savings
100	110	29,441	25,648	12.88%
100	140	27,895	25,324	9.21%
100	170	28,940	27,041	6.56%
100	200	29,198	27,215	6.79%
200	220	77,140	70,891	8.10%
200	280	60,845	59,153	2.78%
200	340	57,766	61,262	-6.05%
200	400	62,348	66,400	-6.49%
400	440	251,243	240,441	4.29%
400	560	175,386	158,377	9.69%
400	680	156,039	153,026	1.93%
400	800	149,617	160,970	-7.58%

Table 3.22: CPU time results for test problem 9.

Chapter 4

CONCLUSIONS

In this thesis, we developed the E^2 Bathe subspace iteration method for solving large, generalized, symmetric-definite eigenvalue problems when several hundred of the least dominant eigenpairs are sought. The E^2 method is a new extension of the basic subspace iteration method (see reference [2]) and also an extension of the recently proposed enriched subspace iteration method (see reference [18]). The E^2 method is able to accelerate the asymptotic convergence rate of the basic and enriched methods by replacing less effective iteration vectors with turning vectors and our newly-developed turning-of-turning vectors.

Compared to the enriched method, the additional computational effort of the E^2 method is due to determining the turning-of-turning vectors. However, this extra computational cost is relatively small compared to the more computationally expensive inverse iterations that the basic, enriched, and E^2 methods each perform. We note that the E^2 method does not require any more inverse iterations than either the basic or the enriched methods. Specifically, for all three methods, the number of inverse iterations that are performed in each iteration of the algorithm is given by $q - p_{k-1}$, where q is the dimension of the subspace and p_{k-1} is the number of converged iteration vectors at the beginning of the k^{th} iteration.

Additionally, we conducted a simplified convergence analysis for the E^2 method and derived the following asymptotic convergence rates corresponding to the i^{th} iteration

vector:

$$\begin{aligned} \boldsymbol{x}_i^{(k)} &\rightarrow \boldsymbol{\phi}_i \text{ at a rate on the order of } \left(\frac{\omega_i^2}{\omega_{q+1}^2}\right)^3 \\ \omega_i^{2,(k)} &\rightarrow \omega_i^2 \text{ at a rate on the order of } \left(\frac{\omega_i^2}{\omega_{q+1}^2}\right)^6 \end{aligned} \tag{4.1}$$

An important conclusion from the convergence analysis in Section 3.3 was that in each iteration, no more than $\frac{q-p_{k-1}}{3}$ iteration vectors will converge at the rate given in (4.1).

The illustrative solutions provided in Section 3.4 demonstrated the improved asymptotic convergence rate that the E^2 method achieves. Additionally, the example results revealed that the iteration vectors converge at a slower, sub-asymptotic rate when the current subspace is close to the target subspace. This is because the current subspace does not turn very much once convergence to the target subspace is nearly achieved. As a result, fewer turning vectors and turning-of-turning vectors are used, which causes the convergence rate for many of the iteration vectors to slow down in the final iterations. We recommend that future acceleration strategies account for this observation.

We also examined the CPU time required for the E^2 method to reach convergence for several three-dimensional test structures. These CPU time results were compared to results obtained using the enriched method to solve the same eigenvalue problems. As indicated in Section 3.4.3, we found many instances in which the E^2 method provided savings in CPU time compared to the enriched method. These savings were often observed when q was in the interval $p < q < \frac{3p}{2}$. However, for $q = 2p$, we found that the enriched method was occasionally able to converge in less time than the E^2 method. Based on this observation, we conclude that there are situations in which the extra computational effort required to perform the second enrichment of the E^2 method is not necessarily offset by the resulting improvement of the asymptotic convergence rate that the E^2 method provides.

While the example problems showed a few instances in which the enriched method converged faster than the E^2 method, there were still many example problems in which the E^2 method out-performed the enriched method. Consequently, we believe that the E^2 method is a useful algorithm for solving the generalized eigenvalue problem, and we

believe that the E^2 method is a strong candidate for future research and improvement. Further research on the E^2 method could be done in the following areas:

- Implementing a parallel version of the algorithm.
- Applying an over-relaxation or shifting strategy to the E^2 method to accelerate convergence even further. These methods have already been adapted to the basic subspace iteration method and could easily be integrated into the E^2 method.
- Developing a hybridization of the basic, enriched, and E^2 methods that can adaptively choose in each iteration which of the three methods to use for that iteration depending on how the subspace is currently converging.
- Adaptively selecting a new q for each iteration depending on how the iteration vectors are currently converging.
- Changing the tolerances in each iteration that are used to determine if a particular iteration vector has turned enough to either be used as a turning vector or a turning-of-turning vector. Reducing this tolerance in the final iterations could mitigate the slow-down effect observed in the convergence rates for the iteration vectors that are close to reaching convergence.

The E^2 method appears to be a useful technique for accelerating the convergence of the Bathe subspace iteration method, and it has the potential to be improved further by investigating the topics that are listed above.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] R.R. Arnold, R.L. Citerley, M. Chargin, and D. Galant. Application of Ritz vectors for dynamic analysis of large structures. *Computers and Structures*, 21(3):461–467, 1985.

[2] Klaus-Jürgen Bathe. *Solution Methods for Large Generalized Eigenvalue Problems in Structural Engineering.* PhD thesis, University of California, Berkeley, 1971.

[3] Klaus-Jürgen Bathe. Convergence of subspace iteration. *Formulations and numerical algorithms in finite element analysis*, pages 575–598, 1977.

[4] Klaus-Jürgen Bathe. The subspace iteration method - revisited. *Computers and Structures*, 126:177–183, 2013.

[5] Klaus-Jürgen Bathe. *Finite Element Procedures.* K.J. Bathe, Watertown, MA, 2014.

[6] Klaus-Jürgen Bathe and Seshadri Ramaswamy. An accelerated subspace iteration method. *Computer Methods in Applied Mechanics and Engineering*, 23:313–331, 1980.

[7] Klaus-Jürgen Bathe and Edward L. Wilson. Large eigenvalue problems in dynamic analysis. In *Journal of the Engineering Mechanics Division Proceedings of the American Society of Civil Engineers*, volume 98, pages 1471–1485, 1972.

[8] Klaus-Jürgen Bathe and Edward L. Wilson. Solution methods for eigenvalue problems in structural mechanics. *International Journal For Numerical Methods Engineering*, 6:213–226, 1973.

[9] Klaus-Jürgen Bathe and Edward L. Wilson. *Numerical methods in finite element analysis.* Prentice-Hall, 1976.

[10] D. Calvetti, L. Reichel, and D.C. Sorensen. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2:1–21, 1994.

[11] Qiang Cui and Ivet Bahar, editors. *Normal Mode Analysis: Theory and Applications to Biological and Chemical Systems*, chapter 6. Chapman and Hall/CRC, 2005.

[12] Jane Cullum and Ralph A. Willoughby. A survey of Lanczos procedures for very large real 'symmetric' eigenvalue problems. *Journal of Computational and Applied Mathematics*, pages 37–60, 1985.

[13] Franciszek A. Dul and Krzysztof Arczewski. The two-phase method for finding a great number of eigenpairs of the symmetric or weakly non-symmetric large eigenvalue problems. *Journal of Computational Physics*, 111:89–109, 1994.

[14] Thomas Ericsson and Axel Ruhe. The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Mathematics of Computation*, 35(152):1251–1268, October 1980.

[15] Mark T. Jones and Merrell L. Patrick. The Lanczos algorithm for the generalized symmetric eigenproblem on shared-memory architectures. *Applied Numerical Mathematics*, pages 377–389, 1993.

[16] Hyung-Jo Jung, Man-Cheol Kim, and In-Won Lee. An improved subspace iteration method with shifting. *Computers and Structures*, 70:625–633, 1999.

[17] Ki-Tae Kim. *The Enriched Subspace Iteration Method and Wave Propagation Dynamics with Overlapping Finite Elements.* PhD thesis, Massachusetts Institute of Technology, June 2018.

[18] Ki-Tae Kim and Klaus-Jürgen Bathe. The Bathe subspace iteration method enriched by turning vectors. *Computers and Structures*, 186:11–21, 2017.

[19] J.D. Kress, G.A. Parker, R. T Pack, B.J. Archer, and W.A. Cook. Comparison of Lanczos and subspace iterations for hyperspherical reaction path calculations. *Computer Physics Communications*, 53:91–108, 1989.

[20] Cornerlius 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*, 45(4):255–282, 1950.

[21] C.C. Paige. Computational variants of the Lanczos method for the eigenproblem. *IMA Journal of Applied Mathematics*, 10(3):373–381, December 1972.

[22] C.C. Paige. Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear Algebra and Its Applications*, 34:235–258, 1980.

[23] Reza Sharifi Sedeh, Mark Bathe, and Klaus-Jürgen Bathe. The subspace iteration method in protein normal mode analysis. *Journal of Computational Chemistry*, 31(1):66–74, 2010.

[24] Horst D. Simon. Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Linear Algebra and Its Applications*, 61:101–131, 1984.

[25] Xuelin Wang and Ji Zhou. An accelerated subspace iteration method for generalized eigenproblems. *Computers and Structures*, 71:293–301, 1999.

[26] Edward L. Wilson, Ming-Wu Yuan, and John M. Dickens. Dynamic analysis by direct superposition of Ritz vectors. *Earthquake Engineering & Structural Dynamics*, 10(6):813–821, 1982.

[27] Kesheng Wu and Horst Simon. A parallel Lanczos method for symmetric generalized eigenvalue problems. *Computing and Visualization in Science*, 2(1):37–46, 1999.

[28] Qian-Cheng Zhao, Pu Chen, Wen-Bo Peng, Yu-Cai Gong, and Ming-Wu Yuan. Accelerated subspace iteration with aggressive shift. *Computers and Structures*, 85:1562–1578, 2007.