The FEAST Algorithm for Sparse Symmetric Eigenvalue Problems

Susanne Bradley *

Department of Computer Science, University of British Columbia

Abstract

The FEAST algorithm is a method for computing extreme or interior eigenvalues of a matrix or matrix pencil. Each step of this algorithm involves repeatedly solving linear systems (which are complex and generally ill-conditioned) along a contour in the complex plane. Most of the previous research done on the FEAST algorithm has used direct solvers for these systems. In this report, we consider instead using iterative solvers for FEAST. Focusing on the standard symmetric eigenvalue problem, we explore the use of MINRES, as well as GMRES with and without preconditioning. Through numerical experiments, we conclude that FEAST can be employed effectively with a preconditioned iterative solver to find eigenpairs of large, sparse matrices.

1 Introduction

Consider the generalized eigenvalue problem, which is to find a scalar λ and an n-dimensional vector $\mathbf{x}$ such that

$$\mathbf{Ax} = \lambda\mathbf{Bx}$$

for $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n\times n}$. If $\mathbf{B}$ is nonsingular, this is equivalent to computing an eigenpair of the matrix $\mathbf{M} := \mathbf{B}^{-1}\mathbf{A}$. The FEAST method [14] finds all such eigenpairs that have λ in some region Γ in the complex plane.

The computational backbone of FEAST is the solution of several linear systems along a contour in the complex plane. If $\mathbf{A}$ and $\mathbf{B}$ are large and sparse, the current implementation of FEAST solves these systems using the Intel MKL-Pardiso solver [15]. Despite the availability of this option, much of the previous work in the FEAST literature has focused on matrices of size 500 to 4,000 [10, 11, 22, 21]. Some exceptions are: the original FEAST paper [14], which employed FEAST on a matrix of size 12,450; the work of and Güttel et al. [9], which had a maximum matrix size of 176,622; and Aktulga et al. [1], which explored FEAST for standard eigenvalue problems of dimensions as high as 327,000. To our knowledge, the only work that has explored the use of iterative solvers in FEAST is that of Krämer et al. [11], which performed empirical testing of different variants of FEAST. In it, GMRES was employed with different tolerances on a matrix of size 1,059.

The purpose of this project is to implement FEAST in MATLAB with iterative linear solvers and explore what impact this has on the performance of the method. We focus primarily on the standard eigenvalue problem (i.e. $\mathbf{B}$ is the identity) with real symmetric $\mathbf{A}$.

*Email: smbrad@cs.ubc.ca

1.1 Overview of the FEAST method

We now describe the derivation of the FEAST algorithm in the case where $\mathbf{A}$ is Hermitian and $\mathbf{B}$ is Hermitian positive definite (for a derivation in the non-Hermitian case, see [21]).

The main computational work of FEAST lies in applying a rational function to $\mathbf{M}$ that approximates the indicator function

$$\pi(\mu) = \frac{1}{2\pi\imath}\int_\Gamma \frac{1}{\gamma-\mu}\mathrm{d}\gamma, \tag{1}$$

where Γ is a contour in the complex plane that encloses the desired eigenvalues. Cauchy's integral theorem tells us that $\pi(\mu) = 1$ for $\mu \in \Gamma$, and $\pi(\mu) = 0$ otherwise.

Define $g(\mu; z) := \frac{1}{z-\mu}$ for a scalar input μ and parameter z. By convention, we define the function $g(\mathbf{M}; z)$ for a matrix $\mathbf{M}$ as $g(\mathbf{M}; z) = (z\mathbf{I} - \mathbf{M})^{-1}$. If we assume that $\mathbf{M}$ is diagonalizable – as is the case with Hermitian $\mathbf{A}$ and $\mathbf{B}$ – we can write $\mathbf{M} = \mathbf{X}\Lambda\mathbf{X}^{-1}$, meaning that

$$\begin{aligned} g(\mathbf{M}; z) :&= (z\mathbf{I} - \mathbf{M})^{-1} \\ &= (z\mathbf{X}\mathbf{X}^{-1} - \mathbf{X}\Lambda\mathbf{X}^{-1})^{-1} \\ &= \mathbf{X}(z\mathbf{I} - \Lambda)^{-1}\mathbf{X}^{-1} \\ &= \mathbf{X}g(\Lambda; z)\mathbf{X}^{-1}, \end{aligned} \tag{2}$$

where $g(\Lambda; z)$ is the diagonal matrix with each diagonal entry λ of Λ replaced by $g(\lambda; z)$.

Our focus is on Hermitian matrices, which have real eigenvalues. Therefore, we assume that our interval of interest is given by $\mathcal{I} := [\lambda_-, \lambda_+] \in \mathbb{R}$. For some complex contour Γ enclosing $\mathcal{I}$, we estimate the integral $\pi(\mu)$ by numerical quadrature. We assume we have m quadrature nodes in the upper half of the complex plane, and m nodes in the lower half. We therefore wish to construct a function of the form

$$\rho_m(\mu) := \sum_{i=1}^{2m} \frac{w_i}{z_i - \mu},$$

with $\rho_m(\mu) \approx \pi(\mu)$. For matrix input, the function is defined as

$$\begin{aligned} \rho_m(\mathbf{M}) :&= \sum_{i=1}^{2m} w_i(z_i\mathbf{I} - \mathbf{M})^{-1} \\ &= \sum_{i=1}^{2m} w_i(z_i\mathbf{B} - \mathbf{A})^{-1}\mathbf{B}. \end{aligned} \tag{3}$$

From Equation (2), we see that $\rho_m(\mathbf{M})$ will have the same eigenvectors $\mathbf{x}_i$ as $\mathbf{M}$, but its eigenvalues will be given by $\rho_m(\lambda_i)$, where λ_i are the original eigenvalues of $\mathbf{M}$. Rather than computing $\rho_m(\mathbf{M})$ directly, we postmultiply by a set of vectors $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_p]$ to compute $\rho(\mathbf{M})\mathbf{Q}$. This entails solving $2m$ linear systems, each with multiple right-hand sides $\mathbf{B}\mathbf{q}_j$. If ρ_m is a good approximation to the indicator function π (Equation (1)), $\rho(\mathbf{M})\mathbf{Q}$ approximates the application of the spectral projector to $\mathbf{Q}$. Specifically, we will have $\rho_m(\mathbf{M})\mathbf{Q} \approx \mathbf{X}_\Gamma\mathbf{X}_\Gamma^H\mathbf{B}\mathbf{Q}$, where $\mathbf{X}_\Gamma$ are the ($\mathbf{B}$-orthogonal) eigenvectors corresponding to the eigenvalues of $\mathbf{M}$ in Γ. In FEAST, we use the vectors resulting from the approximate spectral projection in a Rayleigh-Ritz procedure, and the resulting Ritz vectors form the right-hand side vectors $\mathbf{Q}$ for the

next iteration.

Pseudocode for the FEAST method (as presented in [22]) is given in Algorithm 1. Constructing the filter function ρ_m and selecting a search space dimension p are discussed in Sections 3 and 4, respectively.

Algorithm 1: FEAST

Input: Hermitian $\mathbf{A}$, h.p.d. $\mathbf{B}$, interval $\mathcal{I} = [\lambda_-, \lambda_+]$, search space dimension p
Output: All eigenpairs $(\lambda, \mathbf{x})$ of $\mathbf{M} = \mathbf{B}^{-1}\mathbf{A}$ such that $\lambda \in \mathcal{I}$

Construct $\rho_m(\mu) := \sum_{i=1}^{2m} \frac{w_i}{z_i - \mu}$ to approximate the indicator function for $\mathcal{I}$;
Pick p random n-vectors $\mathbf{Q}_{(0)} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_p]$;
for $k = 1, 2, ...$ *until convergence* **do**
    $\mathbf{Y}_{(k)} \leftarrow \rho(\mathbf{M}) \cdot \mathbf{Q}_{(k-1)}$ (see Equation (3));
    `// Rayleigh-Ritz procedure:`
    Obtain reduced $p \times p$ system: $\hat{\mathbf{A}}_{(k)} \leftarrow \mathbf{Y}_{(k)}^H \mathbf{A} \mathbf{Y}_{(k)}$, $\hat{\mathbf{B}}_{(k)} \leftarrow \mathbf{Y}_{(k)}^H \mathbf{B} \mathbf{Y}_{(k)}$;
    Solve $\hat{\mathbf{A}}_{(k)} \hat{\mathbf{X}}_{(k)} = \hat{\mathbf{B}}_{(k)} \hat{\mathbf{X}}_{(k)} \hat{\Lambda}_{(k)}$ for $\hat{\mathbf{X}}_{(k)}, \hat{\Lambda}_{(k)}$;
    Set $\mathbf{Q}_{(k)} \leftarrow \mathbf{Y}_{(k)} \hat{\mathbf{X}}_{(k)}$;
end

2 Survey of other eigenvalue solvers

In this section we will go over some other established methods for eigenvalue problems.

2.1 Standard eigenvalue problem

To compute the entire spectrum of a (generally fairly small) real matrix, a common method is the QR algorithm [7], which computes the Schur decomposition of a given matrix. Namely, we factor $\mathbf{A} = \mathbf{Q}^H \mathbf{T} \mathbf{Q}$, where $\mathbf{Q}$ is orthonormal (or unitary, if $\mathbf{A}$ is complex), and $\mathbf{T}$ is upper triangular. Here the eigenvalues of $\mathbf{A}$ are given by the diagonal entries of $\mathbf{T}$, and the columns of $\mathbf{Q}$ are called the *Schur vectors* of $\mathbf{A}$.

Sometimes, computing a full Schur factorization is too expensive, or we aren't interested in the entire spectrum of a matrix. Several methods compute only a subset of the spectrum. The simplest of these is the power method [17, chapter 4]: given a nonzero initial vector $\mathbf{v}_0$, the sequence $\mathbf{A}^k \mathbf{v}_0$ will eventually converge to the eigenvector corresponding to the largest modulus eigenvalue λ_1 of $\mathbf{A}$, provided that λ_1 is semi-simple. We can generalize this to subspace iterations: instead of a single vector, we consider the sequence $\mathbf{X}_{(k)} := \mathbf{A}^k \mathbf{X}_{(0)}$, where $\mathbf{X}_{(0)}$ is a system of vectors $[\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_m]$. Under a few assumptions, the columns of $\mathbf{X}_{(k)}$ converge in direction to the m dominant Schur vectors of $\mathbf{A}$ (see [17, chapter 5]). For this to occur, we must force the columns of $\mathbf{X}_{(k)}$ to be linearly independent – otherwise, all columns will converge to the single dominant eigenvector. This is done by computing a QR decomposition $\mathbf{X}_{(k)} = \mathbf{QR}$ at each step and setting $\mathbf{X}_{(k)} := \mathbf{Q}$, which ensures that the vectors will be orthogonal.

Rayleigh-Ritz methods Power methods and subspace iterations are rarely used alone: they more often form a building block for more sophisticated algorithms. A large class of methods uses the Rayleigh-Ritz procedure [17, chapter 4]: given an orthonormal basis $\mathbf{V}_{n \times s}$ that approximates some portion of the spectrum of $\mathbf{A}$, we compute the eigenpairs of a smaller $(s \times s)$ matrix $\mathbf{H} := \mathbf{V}^H \mathbf{A} \mathbf{V}$. If $(\lambda, \mathbf{x})$ is an eigenpair

of $\mathbf{H}$ then $(\lambda, \mathbf{Vx})$ is called a *Ritz pair* of $\mathbf{A}$. If the range of $\mathbf{V}$ approximates the range of the eigenvectors of interest sufficiently well, the Ritz pairs will be a good approximation of the eigenpairs of the original matrix.

The best-known methods of this type are the Arnoldi and Lanczos methods, in which $\mathbf{V}_s$ is an orthonormal basis for the Krylov subspace $\mathcal{K}_s := \text{span}\{\mathbf{v}_0, \mathbf{A}\mathbf{v}_0, \mathbf{A}^2\mathbf{v}_0, ..., \mathbf{A}^{s-1}\mathbf{v}_0\}$ for some nonzero initial vector $\mathbf{v}_0$. This results in an upper Hessenberg matrix $\mathbf{H}_s$ (or tridiagonal, in the case of the Lanczos method). In general, the Ritz pairs will approximate the extremal eigenpairs of $\mathbf{A}$ fairly quickly, but interior eigenpairs will converge more slowly. Some methods – for example, those of Davidson [3] and Jacobi-Davidson [20] – apply preconditioning to the search vectors obtained in an Arnoldi procedure, which can speed up convergence for some matrices.

Filtering methods Computing interior eigenvalues is more challenging. The most conceptually straightforward technique is the shift-and-invert iteration, in which we perform subspace or Krylov iterations using $(\mathbf{A} - \sigma\mathbf{I})^{-1}$ instead of $\mathbf{A}$. This matrix has the same eigenvectors as $\mathbf{A}$, but its eigenvalues are $\lambda_i^{\text{new}} = 1/(\lambda_i - \sigma)$, $i = 1, ..., n$. Hence, the largest-magnitude eigenvalues λ_i^{new} of the inverted matrix will be those corresponding to λ_i near σ. We then have faster convergence of eigenpairs near σ, but at the cost of multiple linear system solves with a matrix that will generally be indefinite, and may be ill-conditioned if $\mathbf{A}$ has eigenvalues very close to σ.

For matrices that are too large to invert, it's common to use filtering techniques that eliminate portions of the search space in the direction of unwanted eigenvectors. For example, one can replace the standard power iterations with polynomial iterations of the form $z_q = p_q(\mathbf{A})z_0$, with p_q being a degree-q polynomial constructed based on some knowledge of the spectrum of $\mathbf{A}$, according to what portion of the spectrum is of interest. Examples of this class of method include the explicitly and implicitly restarted Arnoldi and Lanczos methods [8], and the polynomial filtered Lanczos method developed in [6].

An alternative to polynomial filtering is to construct filters based on rational functions of $\mathbf{A}$ [9]. FEAST is an example of this class of method, as its subspace iterations use a rational function ρ that approximates the indicator function over the interval of interest [22]. Within the same group is the Contour Integral Rayleigh-Ritz (CIRR) method [19]. Like FEAST, this method uses rational functions derived from Cauchy's integral theorem to find eigenpairs in a certain interval; but CIRR instead uses these functions to approximate moment matrices, which are then used in a Rayleigh-Ritz procedure.

2.2 Generalized eigenvalue problem

Of the Rayleigh-Ritz techniques listed above for standard eigenvalue problems, only FEAST and CIRR are directly applicable to the generalized eigenvalue problem. In FEAST in the generalized case, we use numerical integration to estimate the integral

$$\pi(\mathbf{M}) := \frac{1}{2\pi\imath}\int_\Gamma (z\mathbf{B} - \mathbf{A})^{-1}\mathbf{B}\mathbf{q}_j \, \mathrm{d}z,$$

for different right-hand sides $\mathbf{q}_j$. The CIRR method is similar, but instead approximates several integrals of the form

$$s_k := \frac{1}{2\pi\imath}\int_\Gamma (z-\gamma)^k (z\mathbf{B} - \mathbf{A})^{-1}\mathbf{B}\mathbf{q}_j \, \mathrm{d}z, \quad k = 0, 1, ..., m-1,$$

for some γ inside Γ. Note that in FEAST we only approximate the integral s_0. For details and explanation of CIRR, we refer to [18, 19].

3 Numerical quadrature

In this section we focus on computing quadrature nodes and weights – which we will denote by z_i and w_i, respectively – so that

$$\rho_m(\mu) = \sum_{i=1}^{2m} \frac{w_i}{z_i - \mu} \tag{4}$$

approximates the indicator function (as defined by the contour integral $\pi(\mu)$ in Equation (1)) over the interval $[-1, 1]$. We are assuming without loss of generality that the pencil $(\mathbf{A}, \mathbf{B})$ has been transformed linearly such that the wanted eigenvalues are in $[-1, 1]$. We derive node-weight pairs based on Gaussian and trapezoidal quadrature in sections 3.1 and 3.2, respectively, and discuss in section 4 how the quadrature function affects the convergence of FEAST.

Integral parameterization For a given scaling parameter S, we define a family of ellipses Γ_S intersecting the real line at -1 and 1 by:

$$\Gamma_S = \left\{ \gamma : \gamma = \gamma(\theta) = \frac{Se^{\imath\theta} + S^{-1}e^{-\imath\theta}}{S + S^{-1}}, \theta \in [0, 2\pi) \right\}. \tag{5}$$

For $S = \infty$, this gives the unit circle, while for finite S it gives a vertically flattened ellipse. By variable substitution, we can then evaluate the integral $\pi(\mu)$ by

$$\pi(\mu) = \frac{1}{2\pi\imath} \int_0^{2\pi} \frac{\gamma'(\theta)}{\gamma(\theta) - \mu} \mathrm{d}\theta =: \int_0^{2\pi} f_\mu(\theta) \mathrm{d}\theta, \tag{6}$$

where

$$f_\mu(\theta) = \frac{1}{2\pi} \frac{(Se^{\imath\theta} - S^{-1}e^{-\imath\theta})/(S + S^{-1})}{(Se^{\imath\theta} + S^{-1}e^{-\imath\theta})/(S + S^{-1}) - \mu}.$$

3.1 Gaussian quadrature

As in [14], we use m Gauss quadrature nodes $\theta_i^{(G)}$ on the interval $[0, \pi]$ and another m nodes $\theta_i^{(G)}$ on $[\pi, 2\pi]$, all with corresponding quadrature weights $\omega_i^{(G)}$. The quadrature approximation for $\pi(\mu)$ is then given by

$$\pi(\mu) \approx \sum_{i=1}^{2m} \omega_i^{(G)} f_\mu(\theta_i^{(G)}) =: \rho_m^{(G)}(\mu).$$

We then define the mapped Gauss nodes and weights as

$$z_i^{(G)} = \frac{Se^{\imath\theta_i^{(G)}} + S^{-1}e^{-\imath\theta_i^{(G)}}}{S + S^{-1}}, \quad w_i^{(G)} = \frac{\omega_i}{2\pi} \frac{Se^{\imath\theta_i^{(G)}} - S^{-1}e^{-\imath\theta_i^{(G)}}}{S + S^{-1}},$$

and then write

$$\rho_m^{(G)}(\mu) = \sum_{i=1}^{2m} \frac{w_i^{(G)}}{z_i^{(G)} - \mu}. \tag{7}$$

By construction, the nodes and weights both appear in complex conjugate pairs. Specifically, for $i = 1, ..., m$, $z_i = \overline{z_{m+i}}$ and $w_i = \overline{w_{m+i}}$. Note that, for real μ,

$$\frac{\overline{w}}{\overline{z} - \mu} = \overline{\left(\frac{w}{z - \mu}\right)}.$$

Therefore,

$$\frac{w_{m+i}}{z_{m+i} - \mu} = \overline{\left(\frac{w_i}{z_i - \mu}\right)}, \quad i = 1, ..., m,$$

which means we can rewrite Equation (7) as

$$\rho_m^{(G)}(\mu) = 2 \cdot \mathrm{Re}\left(\sum_{i=1}^{m} \frac{w_i^{(G)}}{z_i^{(G)} - \mu}\right). \tag{8}$$

Analogously, for real symmetric matrices $\mathbf{A}, \mathbf{B}$, and a real matrix $\mathbf{Q}$, we can write

$$\rho_m^{(G)}(\mathbf{M})\mathbf{Q} := \sum_{i=1}^{2m} w_i^{(G)}(z_i^{(G)}\mathbf{B} - \mathbf{A})^{-1}\mathbf{B}\mathbf{Q} = 2 \cdot \mathrm{Re}\left(\sum_{i=1}^{m} w_i^{(G)}(z_i^{(G)}\mathbf{B} - \mathbf{A})^{-1}\mathbf{B}\mathbf{Q}\right).$$

Hence, in the strictly real case, we can perform the quadrature step of FEAST using only m of the $2m$ quadrature nodes. This means the required number of linear system solves is now mp instead of $2mp$, where p is the number of columns of $\mathbf{Q}$.

3.2 Trapezoidal quadrature

The construction of the trapezoidal approximant $\rho_m^{(T)}$ is similar to the Gaussian case, but we begin with the trapezoid rule to estimate $\int_0^{2\pi} f_\mu(\theta)\mathrm{d}\theta$. We use equispaced quadrature nodes $\theta_i^{(T)} = \pi(i - 1/2)/m$ and equal weights $\omega_i^{(T)} = \pi/m$ for $i = 1, ..., 2m$. Defining the mapped nodes $z_i^{(T)}$ and weights $w_i^{(T)}$ from $\theta_i^{(T)}$ and $\omega_i^{(T)}$ in the same way as in the Gaussian case, we can now write

$$\rho_m^{(T)}(\mu) = \sum_{i=1}^{2m} \frac{w_i^{(T)}}{z_i^{(T)} - \mu}. \tag{9}$$

In this case, note that for $i = 1, ..., m$, $z_i^{(T)} = \overline{z_{2m-i}^{(T)}}$ and $w_i^{(T)} = \overline{w_{2m-i}^{(T)}}$. Therefore, we can use the same reasoning as we did in the Gaussian case to write

$$\rho_m^{(T)}(\mu) = 2 \cdot \mathrm{Re}\left(\sum_{i=1}^{m} \frac{w_i^{(T)}}{z_i^{(T)} - \mu}\right) \tag{10}$$

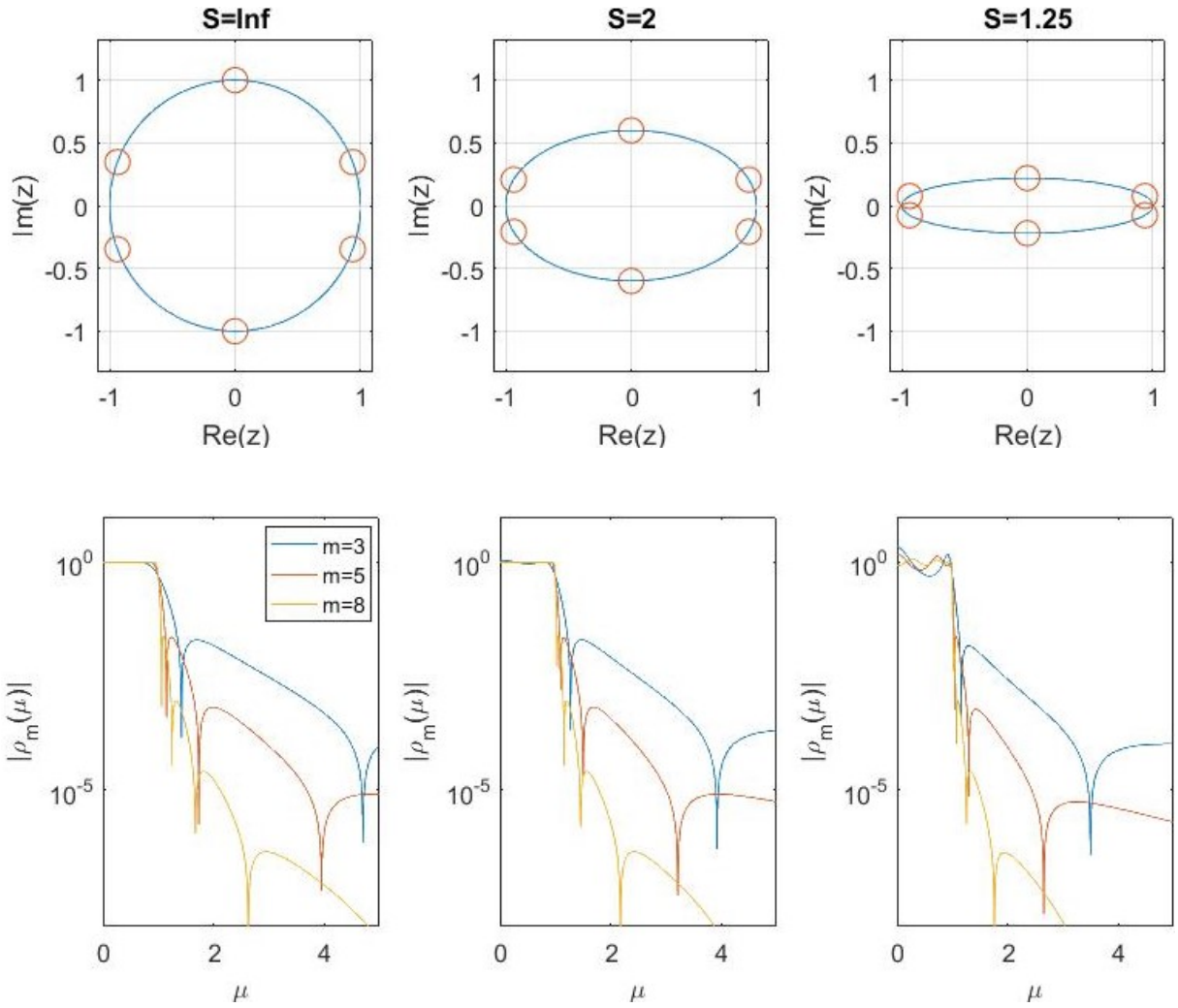


Figure 1: Gaussian quadrature with (half) number of quadrature points $m = 3, 5, 8$, and shape parameters $S = \infty, 2, 1.25$. Top figures show quadrature nodes for $m = 3$, and bottom figures show the value of $\rho(\mu)$ for positive μ only (as ρ is symmetric about $\mu = 0$). Smaller values of S result in more rapid decay outside the interval, at the expense of added oscillations inside the interval.

for all real μ. The analogous result also holds for the matrix case, assuming real symmetric $\mathbf{A}, \mathbf{B}$, and real $\mathbf{Q}$.

4 Convergence of the FEAST method

We now describe how the choice of quadrature function ρ affects convergence of the desired eigenpairs. This will also shed light on how to select a search space dimension p for a particular FEAST instance. We ignore the presence of any errors in the computation of $\rho(\mathbf{M})\mathbf{Q}_{(k)}$. For convergence analysis of FEAST with linear solver errors, we refer the reader to [22].

Assume that our search space has dimension p, and that the eigenvalues $\lambda_j, j = 1, ..., n$ of $\mathbf{M}$ are

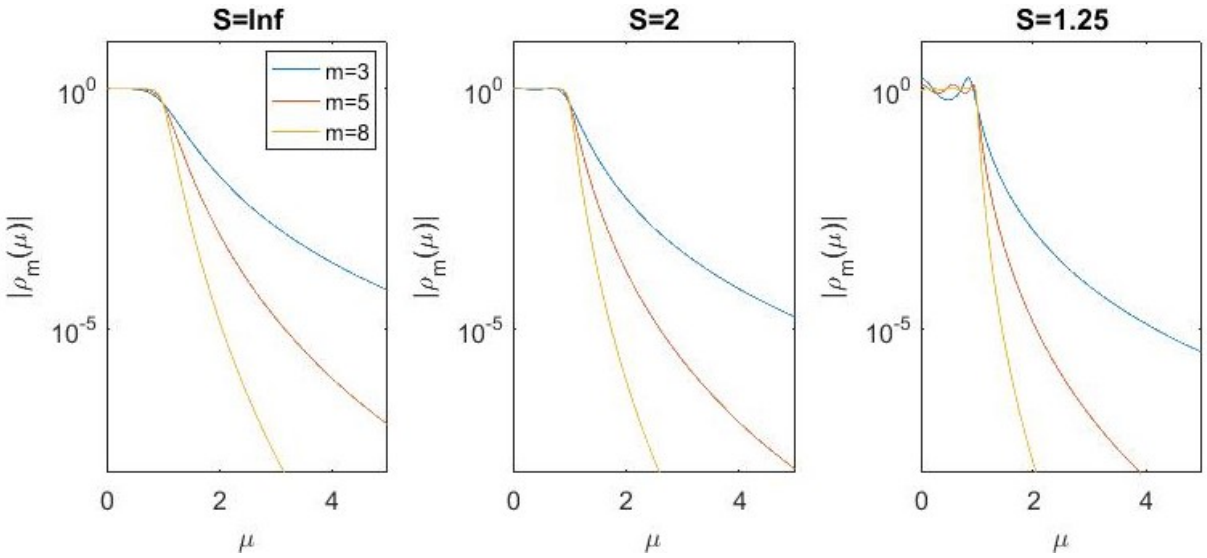


Figure 2: $\rho_m^{(T)}(\mu)$ for the interval $(-1, 1)$ with $S = \infty, 2, 1.25$, and with (half) number of nodes equal to 3, 5, 8. Plots show the results of the integration for real μ. The function is even, so we only show positive μ values. Smaller values of S result in more rapid decay outside the interval, at the expense of added oscillations inside the interval.

ordered such that $|\rho(\lambda_1)| \geq |\rho(\lambda_2)| \geq ... \geq |\rho(\lambda_n)|$. Let $\mathbf{x}_j$ denote the eigenvector corresponding to λ_j. Let $\mathcal{Q}_{(k)} := \text{span}(\mathbf{Q}_{(k)})$, and let q denote the number of eigenvalues of $\mathbf{M}$ in the interval of interest $[\lambda_-, \lambda_+]$. The following result was proven in [22]:

Theorem 1 (Tang and Polizzi, 2014)**.** *At iteration k of FEAST with quadrature rule ρ, for each $j = 1, 2, ..., q$, there is a vector $\boldsymbol{s}_j \in \mathcal{Q}_{(k)}$ such that $||\boldsymbol{x}_j - \boldsymbol{s}_j||_B \leq \alpha|\rho(\lambda_{p+1})/\rho(\lambda_j)|^k$, where α is a constant. In particular, $||(\boldsymbol{I} - \boldsymbol{P}_{(k)})\boldsymbol{x}_j||_B \leq \alpha|\rho(\lambda_{p+1})/\rho(\lambda_j)|^k$, where $\boldsymbol{P}_{(k)}$ is the $\boldsymbol{B}$-orthogonal projector onto the space $\mathcal{Q}_{(k)}$.*

Note that, for the standard eigenvalue problem, the $\mathbf{B}$-norm reduces to the 2-norm. In this case, Theorem 1 tells us that the distance between an eigenvector $\mathbf{x}_j$ and the span of the Ritz vectors is attenuated by a factor of $|\rho(\lambda_{p+1})/\rho(\lambda_j)|$ after each FEAST iteration.

Selecting a search space size From Theorem 1, we see that FEAST will converge more quickly the smaller the value of $|\rho(\lambda_{p+1})|$. For a given quadrature function ρ, this value will be monotonically nondecreasing as the dimension p of our search space is increased. But, because increasing p requires solving additional linear systems at each iteration, there is a trade-off between the FEAST convergence rate and the amount of work per iteration.

In the original FEAST paper [14] and in [22], it was suggested that one should generally use $p = 1.5q$, where q is the number of eigenpairs in the search interval. (This, of course, requires a prior estimate of q, but techniques to address this problem have been developed in [12].) The reasoning behind this recommendation was that, when $\rho(\mu)$ is an 8-point Gaussian quadrature function over a circular contour for the interval $[-1, 1]$, $\rho(1.5) \approx 10^{-3}$. Therefore, if the spectrum of $\mathbf{M}$ is approximately evenly distributed, we would expect to see $||(\mathbf{I} - \mathbf{P}_{(k)})\mathbf{x}_j||_B \to 0$ at a rate of 10^{-3k}.

However, Güttel et al. [9] later explored the use of different quadrature functions with elliptical contours, instead of just circular ones. From Figures 1 and 2, we see that for both quadrature types, functions based on elliptical contours decay more quickly outside the interval of interest, at the expense of some oscillation and early decay within the interval. We would expect – and this was verified experimentally in [9] – that by using elliptical contours, we can achieve a desirable convergence rate with a smaller search space than we could with circular contours.

5 FEAST with iterative linear solvers

The main work of FEAST is in the solution of the linear systems to approximate the spectral projector. For standard problems with real $\mathbf{A}$, these equations are of the form

$$(z_i\mathbf{I} - \mathbf{A})\mathbf{y}_{i,j} = \mathbf{q}_j, \quad i = 1, ..., m,\ j = 1, ..., p. \tag{11}$$

If $\mathbf{A}$ is large and sparse, we would like to be able to solve these equations with iterative solvers. However, the matrices will generally have eigenvalues near the contour points, making the systems ill-conditioned. Additionally, the shifts z_i are complex, meaning that the systems will be non-Hermitian even if $\mathbf{A}$ is symmetric.

5.1 Iterative solver candidates

For each of the p right-hand side vectors $\mathbf{q}_j$, we must solve m shifted systems (with complex shifts). If we use a Krylov solver, we can solve these more efficiently as Krylov subspaces are invariant under shifts – namely, if $\mathbf{V}_s$ is a basis for an s-dimensional Krylov subspace and $\mathbf{H}_s := \mathbf{V}_s^H(-\mathbf{A})\mathbf{V}_s$, then $\mathbf{V}_s^H(z_i\mathbf{I} - \mathbf{A})\mathbf{V}_s = z_i\mathbf{I} + \mathbf{H}_s$, where $\mathbf{H}_s$ is upper Hessenberg.

Symmetric case When $\mathbf{A}$ is symmetric, $\mathbf{H}_s$ is tridiagonal, and therefore so is $z_i\mathbf{I} + \mathbf{H}_s$. Thus, for a given right-hand side $\mathbf{q}_j$, each shifted system reduces to a tridiagonal matrix, all with the same Krylov basis vectors $\mathbf{V}_s$. This means that even though our matrices are non-Hermitian, we can still formulate a three-term recurrence relation between the basis vectors $\mathbf{v}_j$ [16]. Hence, we can solve these systems with the MINRES method [23].

Non-symmetric case When $\mathbf{A}$ is non-symmetric [21], we can still use the same Krylov subspace for the shifted systems, but MINRES is no longer applicable. We can alternatively use the GMRES method. The GMRES-DR-Sh method of [4] is one approach for taking advantage of shifted systems with multiple right-hand sides, although in its original implementation it can't be used with preconditioning.

5.2 Iterative solvers with preconditioning

Ideally, we would like to use a preconditioner $\mathbf{P}$ to accelerate the convergence of the linear solvers we've just described.

MINRES : In general, we can precondition MINRES as long as the preconditioner $\mathbf{P}$ is symmetric positive definite. However, matters are complicated for our problem because the systems are shifted by a complex value z_i. Assume we have factored $\mathbf{P} = \mathbf{P}^{1/2}\mathbf{P}^{1/2}$ (with $\mathbf{P}^{1/2}$ symmetric), and consider the split preconditioned system:

$$\mathbf{P}^{-1/2}(z_i\mathbf{I} - \mathbf{A})\mathbf{P}^{-1/2}(\mathbf{P}^{-1/2}\mathbf{y}_{i,j}) = \mathbf{P}^{-1/2}\mathbf{q}_j.$$

If $(z_i\mathbf{I}-\mathbf{A})$ was real symmetric, it would follow that the iteration matrix $\mathbf{P}^{-1/2}(z_i\mathbf{I}-\mathbf{A})\mathbf{P}^{-1/2}$ would be real symmetric as well. However, in our case the shift z_i is complex, which would mean that the preconditioned matrix, while still being symmetric, will be non-Hermitian. With the complex elements of the system not necessarily restricted to a shift along the diagonal, standard MINRES may not be applicable, as the Krylov basis vectors could no longer be expressed as a three-term recurrence. There are approaches that could be used to overcome this: for example, we could use preconditioning if we used the MINRES variant for complex symmetric systems developed by Choi [2], but an investigation of this is beyond the scope of this report.

GMRES : When we don't precondition GMRES, we can easily take advantage of the shifted systems. Since the Krylov subspaces are invariant under shifts, we can construct a single Krylov basis for each right-hand side.

Matters are more complicated if we wish to use preconditioning. When we precondition the shifted systems, they will no longer be shifted by the identity; thus, their Krylov spaces are no longer equivalent, and reusing information efficiently between the different systems is not as straightforward. There are preconditioned methods that can improve efficiency by taking advantage of related linear systems and multiple right-hand sides – see, for example, the work of Parks et al. [13] – but an exploration of these methods is left as a subject for future work.

6 Numerical experiments

All experiments focus on the standard eigenvalue problem with real symmetric $\mathbf{A}$. We use MINRES-FEAST to denote FEAST with MINRES linear solvers, and GMRES-FEAST to denote the use of GMRES.

All code is implemented in MATLAB. We use the Parallel Computing toolbox with 4 workers in the parallel pool to solve the p different right-hand sides in Equation (11). Experiments were run on a commodity PC with a 3.60 GHz Intel Core i7-4790 processor and 16 GB of memory.

We check for convergence of FEAST by measuring the residual norms of the computed Ritz pairs $(\lambda, \mathbf{x})$ that have $\lambda \in [\lambda_-, \lambda_+]$. Specifically, we say the method has converged if $||\mathbf{Ax} - \lambda\mathbf{x}||_2 < tol$ for all such eigenpairs (with $||\mathbf{x}||_2 = 1$), where tol is a user-specified residual tolerance. The following experiments are all performed with $tol = 10^{-10}$.

6.1 Test matrices

We begin with a brief description of the three real symmetric matrices used in the following experiments. We use the `Trefethen_2000` matrix [1], the Andrews matrix [2] and the `c-big` matrix from the Schenk_IBMNA

[1] `www.cise.ufl.edu/research/sparse/matrices/JGD_Trefethen/Trefethen_2000.html`
[2] `www.cise.ufl.edu/research/sparse/matrices/Andrews/Andrews.html`

group [3], all of which were obtained from the University of Florida Sparse Matrix Collection [5]. Table 1 summarizes some properties of the matrices, and sparsity patterns are shown in Figure 3.

	Matrix		
	`Trefethen_2000`	`Andrews`	`c-big`
n	2,000	60,000	345,241
Nonzeros	41,906	760,154	2,340,859
Full eigen-range	[1.12, 1.74e+04]	[0.0, 36.49]	[−337.19, 338.99]
nnz(L+U), with AMD	1,699,188	234,019,880	144,569,235

Table 1: Summary of the different test matrices used in our numerical experiments.

6.2 GMRES-FEAST vs. MINRES-FEAST

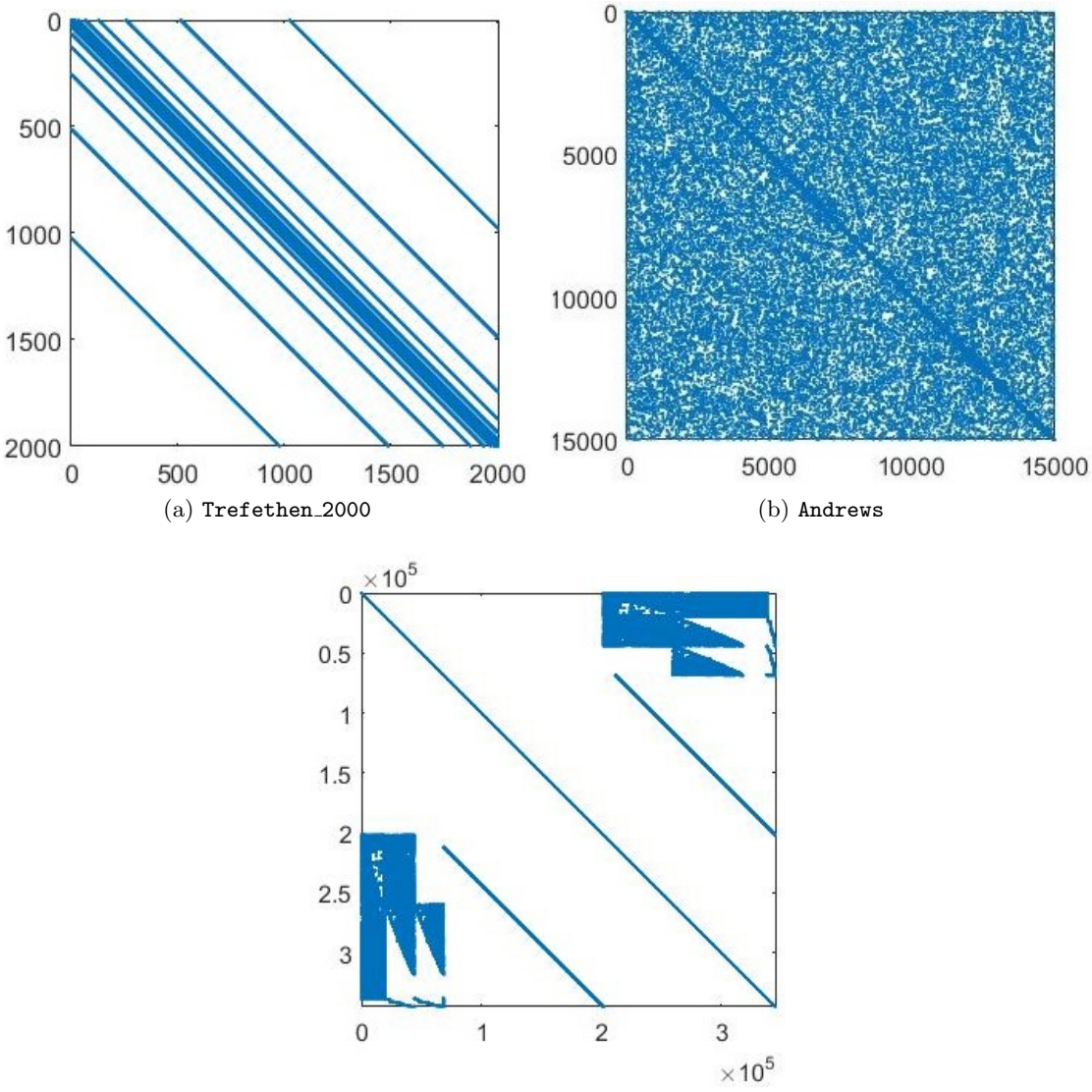
In this experiment we explore the performance of FEAST with MINRES, GMRES, and GMRES preconditioned with ILUTP using a drop tolerance of 0.01. We use no restarting with either GMRES solver. This experiment is performed on the symmetric positive definite 2,000×2,000 `Trefethen_2000` matrix. For our experiment, we find the $q = 20$ eigenpairs with $\lambda \in [31.2, 113.5]$. We use a search space dimension $p = 26$ with the same random initial subspace $\mathbf{Q}_{(0)}$ for all methods. The tolerance of all linear solvers is 10^{-12}. We use 8-point Gaussian quadrature with shape parameter $S = 2.0$. According to Theorem 1, in the absence of linear solver errors this should give us a convergence factor of 4.6×10^{-5} for the slowest-converging eigenpair (note that we can't generally compute this value in practice, as it requires knowledge of the eigenvalues of the matrix).

Results In all cases, FEAST took three iterations to converge. Table 2 summarizes the number of linear solver iterations and computational time required for each iteration.

		Iteration 1 (FEAST)	Iteration 2 (FEAST)	Iteration 3 (FEAST)	Total
MINRES	Solver iterations	166,685	127,213	36,937	330,835
	Time (s)	4.6	3.5	1.2	9.3
GMRES	Solver iterations	146,864	108,305	27,608	282,777
	Time (s)	130.1	99.7	24.7	244.5
GMRES-ILUTP	Solver iterations	1,487	1,513	1,509	4,509
	Time (s)	1.5	1.3	1.3	4.2

Table 2: MINRES-FEAST vs. GMRES-FEAST, for finding 20 eigenpairs of the `Trefethen_2000` matrix. MINRES and GMRES refer to unpreconditioned solvers, with Krylov subspaces recycled between shifted systems, and GMRES-ILUTP refers to GMRES preconditioned with ILUTP($\tau = 0.01$), with all systems solved independently. "Solver iterations" denotes the total number of linear solver iterations used for all $m \cdot p = 208$ linear system solves at each FEAST iteration. Total time for GMRES-ILUTP includes factorization time.

[3] `www.cise.ufl.edu/research/sparse/matrices/Schenk_IBMNA/index.html`



(a) `Trefethen_2000` (b) `Andrews`

(c) `c-big`

Figure 3: Sparsity patterns of the matrices `Trefethen_2000`; (upper left quarter of) `Andrews`; and `c-big`.

At the first iteration – when all solvers have the same random right-hand side vectors – MINRES requires an average of 801 iterations per linear system solve. For GMRES without preconditioning, the average is 706 iterations per linear system solve; however, due to orthogonalization costs, these iterations are much more expensive, and the computational time increases substantially.

We get much better results when we precondition GMRES, instead of taking advantage of the shifts as we do in the unpreconditioned version. Computing the ILUTP factorizations takes a total of 0.05 seconds for all eight systems, and the resulting factorizations $L + U$ have between 2,500 and 2,800 nonzero entries. By preconditioning, we go from an average of 706 iterations per linear system to an average of 7 iterations.

6.3 Preconditioned GMRES-FEAST for large matrices

From the previous experiment, we saw that we gained more computational savings by preconditioning the linear systems than by exploiting symmetry (as with MINRES) or taking advantage of equivalent Krylov subspaces across shifted systems (as with MINRES and unpreconditioned GMRES). We now explore the scalability of preconditioned GMRES-FEAST by testing it on two large matrices: the `Andrews` matrix (size 60,000) and the `c-big` matrix (size 345,241). In both cases, we find 100 eigenpairs (extreme for `Andrews` and interior for `c-big`) using a random initial subspace of size $p = 130$. We use 8-point trapezoidal quadrature with $S = 1.05$. All shifted matrices are preconditioned with ILUTP and all solver tolerances are set to 10^{-10}.

In both cases, convergence of all desired eigenpairs is achieved in three iterations. A summary of the results is given in Table 3. Iteration counts for both the FEAST algorithm and the linear solvers suggest that our approach is effective for large matrices.

	Matrix	
	`Andrews` (60,000×60,000)	`c-big` (345,241×345,241)
Experimental setting		
Interval $[\lambda_-, \lambda_+]$	$[0.0,\ 1.3]$	$[45.2,\ 50.0]$
ILUTP drop tolerance	$5.0 \times 1\text{e}^{-3}$	$1.0 \times 1\text{e}^{-3}$
Results		
Average nnz(L+U) (over all shifts)	2,767,412	5,697,775
Total ILUTP factorization time (s)	1,864.9	60.7
Average GMRES iterations per LS solve	119.0	8.5
Max GMRES iterations for any LS solve	220	16
FEAST iterations to convergence	3	3
Average time per FEAST iteration (s)	8,192.7	2,578.3

Table 3: Results of FEAST for finding 100 eigenpairs of the matrices `Andrews` (size 60,000) and `c-big` (size 345,241).

7 Conclusions

In this report we explored the use of iterative solvers within the FEAST method to compute eigenpairs of sparse symmetric matrices. We conclude that using FEAST with preconditioned GMRES is a promising approach, and we demonstrated the effectiveness of the method by computing eigenpairs of matrices of dimensions up to the hundreds of thousands.

There are several possible ways we might employ iterative solvers even more effectively within FEAST. We observed in our experiments that MINRES iterations are cheap, but that without preconditioning, the iteration counts tend to be rather high. If we apply a complex symmetric version of MINRES, as was developed in [2], we could then use MINRES solvers with preconditioning.

Within FEAST, we solve a few (often eight) different systems, for several right-hand sides. Our preconditioned approach doesn't attempt to exploit any relationship between the different systems, and none of the methods explored here have taken advantage of the fact that each system is being solved

repeatedly for multiple right-hand sides. Krylov subspace recycling methods – particularly those that can be used in conjunction with preconditioning, such as in [13] – represent a promising area of future research.

In performing the numerical experiments, we observed that we require tight linear solver tolerances (generally somewhere between 10^{-9} and 10^{-12}) to obtain reliable FEAST convergence to 10^{-10}. Further research is needed to determine whether there may be ways to reduce the need for very accurate linear system solves. This may also mean that, under some circumstances (such as when we find we need a solver tolerance of 10^{-12}), we gain little from using iterative instead of direct solvers. A question for future research is to determine more precisely the circumstances under which a particular FEAST instance would benefit most from the use of iterative solvers.

A limitation of this work is that it doesn't take full advantage of one of the key benefits of FEAST, which is that FEAST is inherently parallelizable: we can parallelize the solutions of the separate linear systems as we've done here, or we can partition the interval of interest and run FEAST on each segment in parallel (see [1, 9]), or both. We've made limited use of parallelization here, but could achieve better results with more computing resources and more sophisticated parallelization schemes. Another possible direction for future work in this area would be the use of parallel iterative solvers.

References

[1] H. M. Aktulga, L. Lin, C. Haine, E. G. Ng, and C. Yang. Parallel eigenvalue calculation based on multiple shift-invert Lanczos and contour integral based spectral projection method. *Parallel Comput.*, 40(7):195–212, July 2014.

[2] S. T. Choi. Minimal residual methods for complex symmetric, skew symmetric, and skew Hermitian systems. *CoRR*, abs/1304.6782, 2013.

[3] M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson method. *SIAM J. Sci. Comput.*, 15(1):62–76, Jan. 1994.

[4] D. Darnell, R. B. Morgan, and W. Wilcox. Deflated GMRES for systems with multiple shifts and multiple right-hand sides. *Linear Algebra and its Applications*, 429(10):2415 – 2434, 2008.

[5] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1):1:1–1:25, Dec. 2011.

[6] H. Fang and Y. Saad. A filtered Lanczos procedure for extreme and interior eigenvalue problems. *SIAM Journal on Scientific Computing*, 34(4):A2220–A2246, 2012.

[7] J. G. F. Francis. The QR transformation a unitary analogue to the LR transformation, Part 1. *The Computer Journal*, 4(3):265–271, 1961.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[9] S. Güttel, E. Polizzi, P. T. P. Tang, and G. Viaud. Zolotarev quadrature rules and load balancing for the FEAST eigensolver. *SIAM Journal on Scientific Computing*, 37(4):A2100–A2122, 2015.

[10] J. Kestyn, E. Polizzi, and P. T. P. Tang. FEAST eigensolver for non-Hermitian problems. *CoRR*, abs/1506.04463, 2015.

[11] L. Krämer, E. D. Napoli, M. Galgon, B. Lang, and P. Bientinesi. Dissecting the FEAST algorithm for generalized eigenproblems. *CoRR*, abs/1204.1726, 2012.

[12] E. D. Napoli, E. Polizzi, and Y. Saad. Efficient estimation of eigenvalue counts in an interval. *CoRR*, abs/1308.4275, 2013.

[13] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674, 2006.

[14] E. Polizzi. A density matrix-based algorithm for solving eigenvalue problems. *CoRR*, abs/0901.2665, 2009.

[15] E. Polizzi. FEAST eigenvalue solver v3.0: User guide. *CoRR*, abs/1203.4031, 2015.

[16] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[17] Y. Saad. *Numerical methods for large eigenvalue problems*. SIAM, 2011.

[18] T. Sakurai and H. Sugiura. A projection method for generalized eigenvalue problems using numerical integration. *Journal of Computational and Applied Mathematics*, 159(1):119 – 128, 2003. 6th Japan-China Joint Seminar on Numerical Mathematics; In Search for the Frontier of Computational and Applied Mathematics toward the 21st Century.

[19] T. Sakurai and H. Tadano. CIRR: a Rayleigh-Ritz type method with contour integral for generalized eigenvalue problems. *Hokkaido Math. J.*, 36(4):745–757, 11 2007.

[20] G. L. G. Sleijpen and H. A. Van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 17(2):401–425, 1996.

[21] P. T. P. Tang, J. Kestyn, and E. Polizzi. A new highly parallel non-Hermitian eigensolver. In *Proceedings of the High Performance Computing Symposium*, HPC '14, pages 1:1–1:9, San Diego, CA, USA, 2014. Society for Computer Simulation International.

[22] P. T. P. Tang and E. Polizzi. FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. *SIAM Journal on Matrix Analysis and Applications*, 35(2):354–390, 2014. Copyright - 2014, Society for Industrial and Applied Mathematics; Last updated - 2014-04-07.

[23] H. A. Van der Vorst. *Iterative Krylov methods for large linear systems*. Cambridge monographs on applied and computational mathematics. Cambridge University Press, Cambridge, UK, New York, 2003.