Solving the Generalized Symmetric Eigenvalue Problem

Anjos, M.F. and Hammarling, S. and Paige, C.C.

1992

MIMS EPrint: **2008.67**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Unpublished

Solving the Generalized Symmetric Eigenvalue Problem *

M. F. Anjos† S. Hammarling‡ C. C. Paige§

July 2, 1992

Abstract

The generalized symmetric eigenvalue problem (GSEVP) $Ax = \lambda Bx$, A symmetric, B symmetric positive definite, occurs in many practical problems, but there is as yet no numerically stable algorithm which takes full advantage of its structure. The *standard method* (factor $B = GG^T$, solve the ordinary symmetric eigenvalue problem $G^{-1}AG^{-T}(G^Tx) = \lambda(G^Tx)$) is not numerically stable, while the backward stable QZ algorithm takes no account of the special structure of the GSEVP. We show by example a previously unrecognized deficiency in the eigenvectors produced by the QZ algorithm for some cases of this special class of problems. We suggest a new method that takes full advantage of the structure of the GSEVP yet appears not to suffer from either the eigenvector deficiency of the QZ algorithm, nor the lack of accuracy that can be introduced by the standard method when B has a large condition number $\kappa(B)$. The new method first reduces the problem to an equivalent one $A_cy = \lambda D_c^2 y$, A_c symmetric, D_c diagonal. It then implicitly applies Jacobi's method to $D_c^{-1}A_cD_c^{-1}$, while maintaining the symmetric and diagonal forms. The results of numerical tests indicate the benefits of this approach. A variant of the approach is amenable to parallel computation.

Key words: generalized symmetric eigenvalue problem, Cholesky factorization, Jacobi rotations, numerical stability.

AMS Subject Classifications: 65F15, 65F25, 65F35, 65G05

---

*This research was supported by an NSERC of Canada Summer Undergraduate Research Award and by NSERC Grant No. A9236.

†School of Computer Science, McGill University, Montreal, Quebec, Canada, H3A 2A7.

‡The Numerical Algorithms Group, Wilkinson House, Jordan Hill Road, Oxford, UK, OX2 8DR, (nagsven@vax.oxford.ac.uk).

§School of Computer Science, McGill University, Montreal, Quebec, Canada, H3A 2A7, (chris@cs.mcgill.ca).

1 Introduction

In this paper we introduce a new algorithm for the generalized symmetric eigenvalue problem (GSEVP) : Given $A \in \Re^{n\times n}$ such that $A = A^T$ and $B = B^T \in \Re^{n\times n}$ which is positive definite, find $\lambda_1, \ldots, \lambda_n \in \Re$ and linearly independent $x_1, \ldots, x_n \in \Re^n$ which satisfy:

$$Ax_i = \lambda_i B x_i. \tag{1}$$

The approach applies equally well to Hermitian pairs (A, B) with B positive definite. This problem occurs regularly in practical applications, arising for example from discretization of structural problems involving vibrations. The broader class of definite Hermitian pairs (A, B) also has real eigenvalues [3], see [9], but as these can be transformed to problems with B positive definite, and as all the practical problems we have encountered already have B positive definite, we concentrate on the standard GSEVP above.

Our essential motivation is to develop an algorithm which will offer an improvement over other algorithms for those instances of the GSEVP where the matrix B, and possibly also A, is *ill-conditioned* in the sense of having large condition number

$$\kappa(B) \equiv \sigma_1(B)/\sigma_n(B), \qquad \sigma_1(B) \geq \ldots \geq \sigma_n(B)$$

being the singular values of B. In the sequel, we shall use the words "ill-conditioned" when applied to matrices to mean just this. The algorithm is a step towards solving a long standing open problem in matrix computations — how to take advantage of all the structure of the GSEVP while still maintaining all possible numerical stability.

Note that the approach suggested by Crawford [3] can sometimes be used to produce a related GSEVP with a B of improved condition. However if the optimal condition is still poor, we still have to solve the resulting GSEVP accurately. Stewart and Sun [9] give a good discussion of the properties of the GSEVP and its sensitivity.

We begin our exposition by reviewing two current algorithms for solving the GSEVP and characterizing them in terms of their "numerical stability", and the extent to which they preserve structure. This exercise will also allow us to motivate our proposed algorithm. We then proceed by outlining the framework of the proposed algorithm in Section 3. In Section 4 we summarize the numerical tools we shall later apply in the algorithm design. In Section 5 we show how the algorithm can be carried out. Section 6 presents a detailed analysis of the algorithm, showing that it converges in theory, and to what extent it should be numerically reliable. Finally, we conclude by giving numerical results obtained with a FORTRAN implementation of the procedure and a summary of these results. The conclusion also considers the possibility of a parallel implementation of this approach.

2 Two Algorithms and Concepts of Stability

The *standard* algorithm for solving the GSEVP is described in detail in [5, p. 469, Algorithm 8.7.1], see [7, 10] for an early implementation and discussion. It can be briefly summarized in the following way :

- Compute the Cholesky factorization of B, which has the form $B = GG^T$, where G is a lower triangular matrix;
- Form $M = G^{-1}AG^{-T}$ taking full advantage of symmetry;
- Solve the symmetric eigenproblem $My = \lambda y$, whose eigenvalues are mathematically the same as those of the original generalized eigenproblem and whose matrix of eigenvectors Y is related to the matrix of eigenvectors X of the original problem by $Y = G^T X$.

The second step of this method requires solving two systems of equations involving the Cholesky factor G; thus when B, and therefore G, is ill-conditioned, *all* the eigenvalues may incur severe losses in accuracy when they are computed — even those which are relatively insensitive to small perturbations in A and B. The idea is that, in the presence of rounding errors, this algorithm gives G such that $B + \Delta B = GG^T$ which is symmetric, where $\|\Delta B\| \leq c_B u \|B\|$. (We will use $\|\cdot\| = \|\cdot\|_2$ to denote the Euclidean vector norm and the matrix spectral norm, and $\|\cdot\|_F$ the Frobenius norm). Here and later coefficients like c_B will denote some reasonable factor dependent on n alone. The standard algorithm then produces symmetric $M = \mathrm{fl}(G^{-1}AG^{-T})$, and eigenvalues which are exact for symmetric $M + \Delta M$ with $\|\Delta M\| \leq c_M u \|M\|$. The difficulty is that (for exact $GG^T = B$) $\|G^{-1}AG^{-T}\| \leq \|A\|.\|G^{-1}\|^2 = \|A\|.\|B^{-1}\|$ can be very large for ill-conditioned B, and a correspondingly large perturbation ΔM may result in an eigenvalue which is insensitive in the original problem being computed with a large error by this process. Thus even though the standard algorithm is not numerically stable, it does take full advantage of all the structure of the problem.

Another algorithm, the QZ algorithm [8], can also be used to solve the GSEVP. As described in [5, § 7.7], the QZ algorithm computes unitary matrices Q and Z such that :

- $Q^H AZ = T$ is upper triangular;
- $Q^H BZ = S$ is upper triangular.

Since B positive definite implies $s_{ii} \neq 0$, it follows that

$$\lambda_i = t_{ii}/s_{ii}, \qquad i = 1, \ldots, n.$$

Although this method is numerically stable, it has the disadvantage that it does not exploit any of the structure inherent in the GSEVP, namely the symmetry of both A and B and the positive definiteness of B. As our computations in Section 7 will show, the implementation of the QZ algorithm we used can also produce eigenvectors which are inadequate for this specialized problem in a subtle but important sense.

An algorithm is said to be *backward stable* when it always computes the exact solution to a problem "close" (in some sense) to the original problem. The development and clarification of this notion was one of the many important contributions of J.H. Wilkinson to the field of numerical analysis, see for example [10]. Thus on a computer with u denoting the floating

point unit round-off (see [5]), each eigenvalue computed by the QZ algorithm will be exact for the pair $A + \delta A$, $B + \delta B$, where

$$\|\delta A\| \leq c_A u \|A\|, \qquad \|\delta B\| \leq c_B u \|B\|. \tag{2}$$

Note that neither $A + \delta A$ nor $B + \delta B$ is likely to be symmetric, and the computed t_{ii}, s_{ii} and so λ_i will often be complex. Nevertheless, they should be exact for a nearby problem.

Following [2], we can say an algorithm is strongly stable if it preserves the structure of the matrices in the problem *and* it always gives the exact answer to a nearby problem. Thus a strongly stable algorithm for the GSEVP would always produce intermediate symmetric matrices A_c and B_c with the latter being positive definite, and eventually produce answers which are exact for $A + \delta A$ and $B + \delta B$ with the same characteristics, where these equivalent perturbations also satisfy (2). It seems reasonable to consider this notion of stability as the most desirable for a numerical analyst.

Thus while the QZ algorithm is backward stable, and the standard method preserves structure, the motivation for developing and implementing a new approach is to try to obtain both these characteristics, that is, an algorithm for the GSEVP which will satisfy the strong stability criterion. While we have not proven strong stability for our algorithm, we have tried to achieve better accuracy for the solution of a GSEVP with ill-conditioned B through the use of numerical transformations which are as well-conditioned as possible, while ensuring preservation of the problem structure.

3 Algorithm Outline

The algorithm we present consists of two distinct steps, namely :

- Reducing the GSEVP to the special form

  $$A_c y = \lambda D_c^2 y$$

  where A_c is symmetric and D_c^2 is a positive diagonal matrix. The reduction is ideally by well-conditioned transformations, so that A_c has similar condition to the given matrix A, and D_c^2 reflects nearly all the ill-conditioning of B;

- Iterating on this special form to reduce A_c to a diagonal matrix D_A and D_c to a different diagonal matrix D_B while ensuring that the eigenvalues of the original problem will correspond to those of

  $$D_A z = \lambda D_B^2 z.$$

  This final form of the GSEVP can then be solved by simple ratios of the diagonal entries of these matrices, that is $D \equiv D_A D_B^{-2}$ is the matrix of eigenvalues of the GSEVP.

It is important to note that in both steps we must always keep in mind the possible ill-condition of the matrices A and B.

The algorithm also computes an $n \times n$ matrix T of (unnormalized) eigenvectors giving

$$T^T A T = D_A, \qquad T^T B T = D_B^2.$$

The condition of T will be a good indicator of the condition of the transformations used by the algorithm. Our aim is thus to obtain as well-conditioned as possible a T after the algorithm terminates. Note that for the matrix X of normalized eigenvectors

$$X \equiv T D_B^{-1}, \qquad X^T A X = D, \qquad X^T B X = I, \tag{3}$$

so we see $B = (X X^T)^{-1}$, $\kappa(B) = \kappa(X)^2$, and ill-conditioned B will necessarily result in a matrix X of normalized eigenvectors which is never well-conditioned.

4 Numerical Linear Algebra Tools

4.1 Permutation Matrices

A permutation matrix P is the simplest tool we shall use. We will call

$$A_{permuted} = P^T A P$$

a symmetric permutation of A.

4.2 Rotation Matrices

A matrix $Q_{ij} \in \Re^{n \times n}$ is said to be a *rotation* if it is the identity matrix $I_{n \times n}$ except for two diagonal entries, say q_{ii} and q_{jj}, and two off diagonal entries, q_{ij} and q_{ji}, such that, if $i < j$,

$$\begin{aligned} q_{ii} &= q_{jj} = \cos\theta, \\ q_{ji} &= -q_{ij} = \sin\theta, \end{aligned}$$

for some $\theta \in \Re$. We will use these in the following way: Given the vector $(\alpha, \beta)^T$, we design a rotation to "zero" its first component α and put its weight onto the second component β by choosing θ such that

$$\cos\theta = \frac{\beta}{\rho}, \qquad \sin\theta = \frac{\alpha}{\rho},$$

where $\rho = \sqrt{\alpha^2 + \beta^2}$. This yields the desired rotation Q satisfying :

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ \rho \end{pmatrix}.$$

Rotations for n-vectors are built by simply embedding the 2×2 rotation in an $n \times n$ identity matrix, in the obvious way. Our use of rotations stems from their application in the computation of the eigenvalues of symmetric matrices by Jacobi's method [5, § 8.5].

4.3 LDL^T Factorization

We state some results for symmetric matrices. Given $B = B^T \in \Re^{n\times n}$, there exists a unit lower triangular matrix L and a diagonal matrix $\mathcal{D}$ such that

$$B = L\mathcal{D}L^T,$$

if and only if the $n-1$ leading principal submatrices of B are nonsingular [5, § 4.1].

In the particular case where B is also positive definite, all principal submatrices are nonsingular, and thus a unique factorization exists. The $\mathcal{D}$ matrix then has the additional characteristic that all its diagonal entries are strictly positive. Given B symmetric positive definite, we can thus always compute L unit lower triangular and D diagonal such that

$$B = LD^2L^T. \tag{4}$$

It is this latter form of the factorization that we shall use in the first step of our algorithm.

5 Algorithm Description

We now describe how the two steps of Section 3 can be carried out.

5.1 Reducing the GSEVP to the special form

Given a symmetric positive definite matrix $B \in \Re^{n\times n}$, our algorithm computes the factorization (4) of P^TBP for some symmetric permutation of B. Because of the possible ill-condition of B, we will require that the factorization has the following property : The matrix L should be as well-conditioned as possible, and so, intuitively, we may say that all the "ill-condition" should be kept in the matrix D. More specifically, we require L to be computed so that it is unit lower triangular with all its off-diagonal entries relatively small, ideally having (see Section 6)

$$|l_{ij}| < 1, \qquad 1 \le j < i \le n.$$

This property is desirable because it will ensure that L is well-conditioned, and thus it can be used in forming $L^{-1}AL^{-T}$ without risking much loss of accuracy in the computation of the eigenvalues. Although it is in general ineffective to just place the diagonal elements of B in decreasing order prior to the computation of L and D, our objective can be attained by judicious use of permutations as we now describe. The following is a slight variant of the usual Cholesky factorization of B. The order of operations (sometimes called the outer-product formulation) facilitates the choice of pivots later.

When there are no permutations, B is reduced to diagonal form in $n-1$ steps using elementary unit lower triangular matrices L_i in

$$D^2 = L_{n-1}\cdots L_2L_1BL_1^TL_2^T\cdots L_{n-1}^T.$$

Each L_i has the form for $i = 1, \ldots, n-1$,

$$L_i = I - t_i e_i^T, \quad t_i = \begin{pmatrix} 0 \\ l_i \end{pmatrix}, \quad l_i \in \Re^{n-i}, \tag{5}$$

where e_i is the ith column of the $n \times n$ unit matrix.

If we write

$$\begin{pmatrix} D_{1,i-1}^2 & 0 \\ 0 & B_{i,n} \end{pmatrix} \equiv L_{i-1} \cdots L_1 B L_1^T \cdots L_{i-1}^T \tag{6}$$

where $D_{1,i-1}$ is an $i-1 \times i-1$ diagonal matrix, then this matrix is transformed in the ith step via

$$\underbrace{\begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -l_i & I \end{pmatrix}}_{L_i} \begin{pmatrix} D_{1,i-1}^2 & 0 & 0 \\ 0 & \tilde{b}_{ii} & \tilde{b}_i^T \\ 0 & \tilde{b}_i & \tilde{B}_{i+1,n} \end{pmatrix} \underbrace{\begin{pmatrix} I & 0 & 0 \\ 0 & 1 & -l_i^T \\ 0 & 0 & I \end{pmatrix}}_{L_i^T} \tag{7}$$

where taking

$$l_i = \frac{\tilde{b}_i}{\tilde{b}_{ii}}$$

gives this the desired form

$$\begin{pmatrix} D_{1,i}^2 & 0 \\ 0 & B_{i+1,n} \end{pmatrix} \equiv \begin{pmatrix} D_{1,i-1}^2 & 0 & 0 \\ 0 & \tilde{b}_{ii} & 0 \\ 0 & 0 & \tilde{B}_{i+1,n} - \tilde{b}_i \tilde{b}_{ii}^{-1} \tilde{b}_i^T \end{pmatrix}. \tag{8}$$

After $n-1$ such steps $D^2 = D_{1,n}^2$ is diagonal. Since this is effectively the Cholesky factorization, the $\tilde{b}_{ii}$ are positive when B is positive definite [5, § 4.2].

To preserve numerical stability in our transformations of A, we now introduce permutations in order to make $\tilde{b}_{ii}$ the largest diagonal element of $B_{i,n}$, for each i. Hence, we actually compute this factorization as

$$D^2 = L_{n-1} P_{n-1}^T \cdots L_1 P_1^T B P_1 L_1^T \cdots P_{n-1} L_{n-1}^T, \tag{9}$$

where at the ith step a permutation P_i is designed to bring the element of largest magnitude in the diagonal of $B_{i,n}$ to the (i,i) position.

If we apply these transformations to A, the original problem $Ax = \lambda Bx$ becomes

$$\begin{aligned} A_c y &= \lambda D_c^2 y, \\ A_c &\equiv L_{n-1} P_{n-1}^T \cdots L_1 P_1^T A P_1 L_1^T \cdots P_{n-1} L_{n-1}^T, \\ D_c &\equiv D, \\ y &\equiv L_{n-1}^{-T} P_{n-1}^T \cdots L_1^{-T} P_1^T x. \end{aligned} \tag{10}$$

Thus when we have found an eigenvector y of (10), the eigenvector x of (1) is

$$x = T_c y, \qquad T_c \equiv P_1 L_1^T \cdots P_{n-1} L_{n-1}^T, \tag{11}$$

and computationally we can transform A to A_c and produce T_c as we reduce B to diagonal form. Note that

$$A_c = T_c^T A T_c, \qquad D_c^2 = T_c^T B T_c, \tag{12}$$

where T_c is our matrix of accumulated transformations.

The usual theory, see for example [5, §3.4.4], shows

$$\begin{aligned} B &= P^T \tilde{L} D^2 \tilde{L}^T P, \qquad T_c = P^T \tilde{L}^{-T}, \\ P &\equiv P_{n-1}^T \cdots P_1^T, \qquad \tilde{L} \equiv \tilde{L}_1^{-1} \cdots \tilde{L}_{n-1}^{-1}, \end{aligned} \tag{13}$$

where since $L_i^{-1} = I + t_i e_i^T$, we have $\tilde{L}_i^{-1} = I + \tilde{t}_i e_i^T$ with $\tilde{t}_i \equiv P_{n-1}^T \cdots P_{i+1}^T t_i$ containing the subdiagonal elements of the ith column of $\tilde{L}$. However there does not appear to be any need to keep this $\tilde{L}$ and P separately. If we do not want to compute eigenvectors, we can apply the transformations to produce D_c and A_c as in (9) and (10), and discard each one after it has been used. When we want eigenvectors it appears preferable to compute, and later update, T_c in (11). This is partly because our next step uses transformations which are not triangular.

5.2 Iterating on the special form

We now present the iteration applied to (10) to obtain the final form of the problem

$$D_A z = \lambda D_B^2 z, \tag{14}$$

where both D_A and D_B are diagonal matrices.

This iterative algorithm executes a sequence of sweeps designed to bring the matrix A_c to diagonal form, while keeping D_c diagonal. Here the rotations are not designed solely on the entries of A_c, but rather on the entries of

$$D_c^{-1} A_c D_c^{-1}. \tag{15}$$

We feel direct computation of the product (15) should be avoided, because with ill-condition of D_c much accuracy could be lost when later transformations are applied to (15).

The main idea for this part of the algorithm is to *never* form the product (15) during the iterations. The key observation is that designing and applying a rotation Q to the matrix (15) for the problem (10)

$$Q^T D_c^{-1} A_c D_c^{-1} Q \bar{y} = \lambda \bar{y}, \qquad \bar{y} \equiv Q^T D_c y, \tag{16}$$

actually corresponds to the "transformation" between the two D_c matrices in

$$A_c y = \lambda D_c \underbrace{Q Q^T}_{=I} D_c y = \lambda D_c^2 y. \tag{17}$$

Therefore, at each substep of our algorithm we shall design a rotation Q_{ij} in the i,j-plane, with $i < j$, based on the entries of (15), but we will not *apply* that rotation. Note we do

not suggest the following is the optimal implementation, it is simply the one we chose to experiment with. Following (11), we assume $x = T_c y$ is the corresponding eigenvector of (1), so that T_c is the matrix of transformations to this point.
THE ITERATION :

1. Compute a rotation Q_{ij} such that $Q_{ij}^T D_c^{-1} A_c D_c^{-1} Q_{ij}$ in theory makes the entries at positions (i,j) and (j,i) of the product (15) zero. It is important to note that only three elements of (15) need to be considered here.

2. Choose a diagonal matrix D_c' and form

$$N = D_c^{-1} Q_{ij} D_c'.$$

   D_c' is chosen so N is as well-conditioned as possible. The theory for this is given in Section 6.2.

3. Form

$$A_c^{'} = N^T A_c N,$$

   to again bring the problem to the form

$$A_c^{'} y^{'} = \lambda D_c^{'2} y^{'}, \qquad y^{'} \equiv N^{-1} y.$$

4. Since now $x = T_c N y'$ is the corresponding eigenvector of (1), we update the matrix of eigenvector transformations as well (if we want to compute the eigenvectors)

$$T_c' = T_c N.$$

This process is repeated for all (i,j), $1 \le i < j \le n$, until the sweep has been completed. Note there can be different orderings to cover the range of valid (i,j) pairs within a sweep, and the step need not be applied if elements are already sufficiently small. For testing the algorithm we used the standard ordering by rows.

After each sweep, we expect to have significantly decreased the "weight" of the off-diagonal elements of $D_c^{-1} A_c D_c^{-1}$. Sweeps are repeated until all off diagonal entries of $D_c^{-1} A_c D_c^{-1}$ are effectively zero (i.e. up to a given tolerance below which values are taken to be zero). This is the same termination criterion as used in [6] and was good enough to show this approach works. We would look for a more relevant criterion for the GSEVP in a production code. On convergence the problem is reduced to the form (14) with D_A the limit of the A_c and D_B the limit of the D_c. We also write T as the limit of the T_c.

Finally, for the problem (14) we have

$$D_B^{-1} D_A D_B^{-1} e_i = \lambda_i e_i,$$

so the eigenvalues are

$$\lambda_i = \frac{D_A(i,i)}{D_B(i,i)^2}, \quad 1 \le i \le n, \tag{18}$$

and (1) has corresponding eigenvectors $x_i = Te_i$, that is T is the matrix of unnormalized eigenvectors of the original problem. Since $D_A = T^TAT$, $D_B^2 = T^TBT$, we see $D_B^{-1}T^TBTD_B^{-1} = I$, so $X \equiv TD_B^{-1}$ is our matrix of normalized eigenvectors. This gives $X^TAX = D_B^{-1}D_AD_B^{-1} = D \equiv \text{diag}(\lambda_i)$.

6 Theoretical Analysis

To try to understand why this approach seems to give numerically reliable results, we will first show that the transformations used in the first stage, described in Section 5.1, are well-conditioned, while we suspect those in the second stage, Section 5.2, will not be significantly worse than the condition of the given problem merits. We also show that in theory the process converges, eventually at a quadratic rate. After that we discuss the representation of eigenvalues of the GSEVP in order to understand what to expect from a backward stable algorithm for the GSEVP. We emphasize a particularly effective measure of the accuracy of computed eigenvalues of the GSEVP.

6.1 Analysis of the reduction to the special form

Here we show the L_i transformations in (9), see (10), are well-conditioned. First, we see from (7) that $l_i = \tilde{b}_i/\tilde{b}_{ii}$ comes from the positive definite matrix with Cholesky factorization

$$B_{i,n} = \begin{pmatrix} \tilde{b}_{ii} & \tilde{b}_i^T \\ \tilde{b}_i & \tilde{B}_{i+1,n} \end{pmatrix} = \begin{pmatrix} g_{ii} & 0 \\ g_i & G_{i+1} \end{pmatrix} \begin{pmatrix} g_{ii} & g_i^T \\ 0 & G_{i+1}^T \end{pmatrix}, \quad \text{say.}$$

Thus the jth element l_{ji} of l_i is

$$\tilde{b}_{ji}/\tilde{b}_{ii} = g_{ji}g_{ii}/g_{ii}^2 = g_{ji}/g_{ii}.$$

But with the pivoting strategy used in (9)

$$g_{ii}^2 \geq g_{ji}^2 + e_j^TG_{i+1}G_{i+1}^Te_j > g_{ji}^2,$$

since $B_{i,n}$ is *positive* definite, giving as desired

$$|l_{ji}| = |g_{ji}/g_{ii}| < 1.$$

It follows that every subdiagonal element of L_i is less than 1 in magnitude, as is every subdiagonal element of $\tilde{L}$ in (13), and so these transformations are well-conditioned. In Section 7, we illustrate some of the condition numbers found computationally, and these were all small. As a result it would appear to be a waste of effort on a single processor computer to find the eigendecomposition of B, see [10, p.337], (or the singular value decomposition of G), if these are not already at hand, in order to diagonalize B in this step. Such an approach might however be useful for multiprocessor computers, see Section 8.

6.2 Convergence, and condition of the iteration transformations

In order to maintain the form $A_c y = \lambda D_c^2 y$ with A_c symmetric and D_c diagonal, we consider transformations N giving

$$A_c' y' = N^T A_c N(N^{-1}y) = \lambda N^T D_c^2 N(N^{-1}y) = \lambda D_c'^2 y',$$

such that D_c' remains diagonal. But this is only possible if $D_c N D_c'^{-1}$ is an orthogonal matrix, say Q, so we restrict ourselves to

$$N \equiv D_c^{-1} Q D_c', \qquad QQ^T = Q^T Q = I, \tag{19}$$

for some orthogonal matrix Q and diagonal matrix D_c', giving

$$A_c' = N^T A_c N = D_c' Q^T D_c^{-1} A_c D_c^{-1} Q D_c'.$$

We could choose Q in different ways, but the obvious computational choice is the Jacobi rotation Q_{ij} to zero the i,j and j,i elements of $D_c^{-1} A_c D_c^{-1}$. Since we produce A_c' and D_c' from A_c and D_c where in theory

$$D_c'^{-1} A_c' D_c'^{-1} = Q_{ij}^T D_c^{-1} A_c D_c^{-1} Q_{ij},$$

we see we are implicitly applying Jacobi's method to the $D_c^{-1} A_c D_c^{-1}$ produced by (10), so we have all the theoretical convergence properties of Jacobi's method, including fairly rapid onset of quadratic convergence.

Ideally D_c' should be such that $N \equiv D_c^{-1} Q_{ij} D_c'$ is as well-conditioned as possible, so it will be identical to D_c in all but the i and j elements. To choose these, we need only consider the 2×2 case, where with obvious notation:

$$N = D^{-1} Q D' = \begin{pmatrix} d_1^{-1} & \\ & d_2^{-1} \end{pmatrix} \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} d_1' & \\ & d_2' \end{pmatrix} = \begin{pmatrix} cd_1'/d_1 & -sd_2'/d_1 \\ sd_1'/d_2 & cd_2'/d_2 \end{pmatrix}. \tag{20}$$

For any real 2×2 matrix N with $\phi \equiv \|N\|_F$, $\delta \equiv |\det N|$, it is straightforward to show

$$\begin{aligned} \kappa_F(N) &\equiv \|N\|_F \|N^{-1}\|_F = \phi^2/\delta, \\ \kappa(N) &= \left(\phi^2 + \sqrt{\phi^4 - 4\delta^2}\right)/(2\delta) \\ &= \left[\kappa_F(N) + \sqrt{\kappa_F(N)^2 - 4}\right]/2, \end{aligned} \tag{21}$$

so $\kappa(N)$ clearly has its minimum when $\kappa_F(N)$ does. Thus we only need analyse $\kappa_F(N)$, which is easier than $\kappa(N)$. ¿From (20) we see

$$\begin{aligned} \phi^2 &= \{s^2[(d_1 d_1')^2 + (d_2 d_2')^2] + c^2[(d_1 d_2')^2 + (d_2 d_1')^2]\}/(d_1 d_2)^2, \\ \delta &= detD^{-1}.detD' = (d_1' d_2')^2/(d_1 d_2)^2, \end{aligned}$$

and defining $\rho \equiv d_1/d_2$ and $\zeta \equiv d_1'/d_2'$ we obtain

$$\kappa_F(N) = \phi^2/\delta = [c^2(\rho^2 + \zeta^2) + s^2(\rho^2\zeta^2 + 1)]/(\rho\zeta),$$

which has just one unknown ζ. Taking derivatives with respect to ζ shows this has a minimum when

$$\zeta_{opt}^2 = (s^2 + \rho^2 c^2)/(c^2 + \rho^2 s^2), \tag{22}$$

and at this value,

$$\begin{aligned} \kappa_F(N)_{min} &= 2\sqrt{1 + s^2c^2(\rho - \rho^{-1})^2}, \\ \kappa(N)_{min} &= \sqrt{1 + s^2c^2(\rho - \rho^{-1})^2} + |sc(\rho - \rho^{-1})|. \end{aligned} \tag{23}$$

We see for a given value of ρ that $\kappa_F(N)_{min}$ is its worst for $|sc| = 1/2$, and then $\kappa_F(N)_{min} = \rho + \rho^{-1}$. But when either c or s is small then $\kappa_F(N)_{min}$ is significantly less than the value $\kappa_F(N) = \rho + \rho^{-1}$ given by $D_c' = I$. In many cases sc will be small, especially as convergence proceeds, so this should have an advantage over Jacobi's method applied directly to (15).

Now that we know the optimal ratio $d_1'/d_2' = \zeta_{opt}$, we consider the size of d_2' and so $d_1' = d_2'\zeta_{opt}$. Since at each step $D_c^{-1}A_cD_c^{-1}$ is the result of applying Jacobi steps to the original such form, our choice of N satisfying (19) will not alter these, that is, the future Q_{ij}. Also since the future ζ_{opt} only depend on the c and s from Q_{ij} and the ratio $\rho = d_1/d_2$, we see in theory it makes no difference what scaling we choose. Computationally it is preferable to limit the change in size, so we can insist $\|D_c'\|_F = \|D_c\|_F$, and then

$$d_1^2 + d_2^2 = d_1'^2 + d_2'^2 = (\zeta_{opt}^2 + 1)d_2^2.$$

But $(\zeta_{opt}^2 + 1) = (1 + \rho^2)/(c^2 + s^2\rho^2)$, giving

$$\begin{aligned} d_2^2 &= (d_1^2 + d_2^2)(c^2 + s^2\rho^2)/(1 + \rho^2) = c^2d_2^2 + s^2d_1^2, \\ d_1^2 &= (c^2d_2^2 + s^2d_1^2)(s^2 + c^2\rho^2)/(c^2 + s^2\rho^2) = c^2d_1^2 + s^2d_2^2, \end{aligned}$$

and so

$$N = \begin{pmatrix} c\sqrt{c^2 + s^2/\rho^2} & -s\sqrt{s^2 + c^2/\rho^2} \\ s\sqrt{s^2 + c^2\rho^2} & c\sqrt{c^2 + s^2\rho^2} \end{pmatrix}. \tag{24}$$

If in the Jacobi rotations we choose $s^2 \le c^2$, then d_1' will be closer to d_1, and d_2' to d_2, giving as small a possible change from D_c to D_c' in some sense.

Alternate possibilities for N that save a little computation are

$$N = \begin{pmatrix} \zeta_{opt}/\rho & -\tau/\rho \\ \tau\zeta_{opt} & 1 \end{pmatrix}, \qquad N = \begin{pmatrix} 1 & \tau/\zeta_{opt} \\ \tau\rho & \rho/\zeta_{opt} \end{pmatrix}, \tag{25}$$

where $\tau \equiv s/c$. In all cases it is sensible to form N fully, and then apply it, as this gives the least computation and the most accuracy.

The reader might be concerned that these N matrices can still be ill-conditioned. This is a important concern, and for many years we tried to avoid any such ill-condition. However if we want to *diagonalize* a definite pair, we are forced in general to use such matrices. Our hope is that this algorithm introduces very little ill-condition that is not already inherent in the problem.

6.3 Testing the computed results

We can multiply (1) by any positive number to give

$$\beta_i A x_i = \alpha_i B x_i, \tag{26}$$

and refer to (α_i, β_i) as a generalized eigenvalue (GEN) of (A, B). If a computed GEN (α, β) and its eigenvector x are found using a backward stable algorithm, then

$$\begin{aligned} \beta(A + \delta A)x &= \alpha(B + \delta B)x, \\ \|\delta A\| \leq c_A u \|A\|, &\qquad \|\delta B\| \leq c_B u \|B\|. \end{aligned} \tag{27}$$

This means that

$$\begin{aligned} \|\beta A x - \alpha B x\| &= \| - \beta \delta A x + \alpha \delta B x\| \\ &\leq (|\beta| \|\delta A\| + |\alpha| \|\delta B\|) \|x\| \\ &\leq (|\beta| c_A \|A\| + |\alpha| c_B \|B\|) \|x\| u. \end{aligned}$$

We would expect these upper bounds to be roughly approached by a backward stable algorithm, so to check our computed results, we can compare the following *performance index*

$$\frac{\|A x_c \beta_c - B x_c \alpha_c\|_2}{(|\beta_c| \|A\|_F + |\alpha_c| \|B\|_F) \|x_c\|_2 u}$$

with unity, for each computed GEN (α_c, β_c) and corresponding eigenvector x_c. Note we have used the Frobenius matrix norm to attempt to include some of the effect of c_A and c_B. But for some scalings of the GEN this was found not to be a totally consistent index for problems with a wide range of λ_i. By taking

$$\beta \equiv (1 + \lambda^2)^{-1/2}, \qquad \alpha \equiv \lambda \beta, \qquad \text{so} \quad \alpha^2 + \beta^2 = 1,$$

we found computationally that this was a very consistent and effective performance index.

To test each algorithm consistently, we also formed the normalized matrix X of eigenvectors as in (3), and checked

$$\|AXD_B - BXD_A\|_F, \qquad \|X^T A X - D\|_F, \qquad \|X^T B X - I\|_F,$$

with $D_A = \text{diag}(\alpha_i)$, $D_B = \text{diag}(\beta_i)$. The surprising result was that eigenvectors from the QZ implementation we used sometimes failed to diagonalize A at all well.

7 Numerical Results

This algorithm was implemented in FORTRAN 77 and was tested on several examples of GSEVPs. The iterative process was stopped when all off-diagonal entries of $D_c^{-1} A_c D_c^{-1}$ had

magnitude smaller than $\epsilon = 10^{-16}$ relative to the largest element of $D_c^{-1}A_cD_c^{-1}$, however a better criterion may be desirable.

We include computational results for one significant example. The results of our algorithm are compared with those from two other methods. One is the standard algorithm (DSYGV from Lapack [1]) which computes the Cholesky factor G of B such that $B = GG^T$, G lower triangular, forms $M = G^{-1}AG^{-T}$ and computes the eigenvalues of M, which are mathematically the same as for the original problem (1). The other is the QZ algorithm (using the sequence of routines QZHES, QZIT, QZVAL and QZVEC from Eispack [4]) for the general $Ax = \lambda Bx$ problem, which does not assume any symmetry or positive definiteness of A or B. Since the eigenvalues are in general complex valued, QZVAL returns two components (real and imaginary) for each eigenvalue. After calling QZVAL, our code verifies that | imaginary component | $< \epsilon|$ maximum eigenvalue | for each eigenvalue, a weak condition which was never violated during our tests. In the results below, we list only the real component returned by QZVAL as representing the eigenvalue computed by the QZ algorithm.

We also illustrate the effectiveness of the step of our algorithm that computes the factorization (13) with $\tilde{L}$ very well-conditioned. In our tests we used a variety of B matrices with various conditions. We give here a listing of the condition numbers of a small selection of those symmetric positive definite matrices together with the conditions for the computed $\tilde{L}$ and D^2.

Our computations were performed in FORTRAN 77 using DOUBLE PRECISION variables on a SUN 4 computer for which the machine precision as given by MATLAB is $u = 2.22E - 16$.

TEST EXAMPLE :

This example was constructed by choosing a random orthogonal matrix Q of size 8×8, plus the two diagonal matrices

$$D_a = diag(1, 2, 3, 4, -5, 6, 7, 8),$$
$$D_b = diag(0.0008, 800000, 8, 0.08, 80, 0.00008, 80000, 0.008),$$

and forming $A = QD_aQ^T$, $B = QD_bQ^T$. We thus know, by construction of A and B, that the real eigenvalues are $\lambda_i = D_a(i,i)/D_b(i,i)$ for $i = 1, \ldots, 8$.

Results :

The matrix A with $\kappa(A) = 8.801$ and $\|A\|_F = 14.28$ is :

Columns 1 to 4

$0.7998E+00$	$-0.2007E+01$	$0.4772E+00$	$0.2622E+01$
$-0.2007E+01$	$0.6987E+01$	$0.2478E-01$	$0.1407E-02$
$0.4772E+00$	$0.2478E-01$	$0.6037E+01$	$-0.1937E+01$
$0.2622E+01$	$0.1407E-02$	$-0.1937E+01$	$0.2860E+01$
$-0.3499E+01$	$-0.1390E+00$	$0.1192E+01$	$0.1278E+01$
$-0.1042E+01$	$-0.4810E+00$	$-0.2277E+00$	$-0.3337E+00$
$0.1399E+01$	$0.2929E+00$	$-0.2671E+00$	$-0.6261E+00$
$-0.7488E+00$	$-0.1532E+00$	$0.1473E+00$	$0.3407E+00$

Columns 5 to 8

$$\begin{array}{cccc}
-0.3499E+01 & -0.1042E+01 & 0.1399E+01 & -0.7488E+00 \\
-0.1390E+00 & -0.4810E+00 & 0.2929E+00 & -0.1532E+00 \\
0.1192E+01 & -0.2277E+00 & -0.2671E+00 & 0.1473E+00 \\
0.1278E+01 & -0.3337E+00 & -0.6261E+00 & 0.3407E+00 \\
0.1284E+01 & -0.1885E+01 & 0.8942E+00 & -0.4736E+00 \\
-0.1885E+01 & 0.4672E+01 & -0.6121E+00 & 0.2630E+00 \\
0.8942E+00 & -0.6121E+00 & 0.2032E+01 & -0.5794E+00 \\
-0.4736E+00 & 0.2630E+00 & -0.5794E+00 & 0.1327E+01
\end{array}$$

The matrix B with $\kappa(B) = 1.229E+10$ and $\|B\|_F = 8.040E+05$ is :
Columns 1 to 4

$$\begin{array}{cccc}
0.4612E+02 & -0.3923E+03 & 0.8760E+03 & 0.6374E+02 \\
-0.3923E+03 & 0.1712E+05 & -0.2764E+05 & 0.1868E+05 \\
0.8760E+03 & -0.2764E+05 & 0.6793E+05 & 0.2018E+05 \\
0.6374E+02 & 0.1868E+05 & 0.2018E+05 & 0.1291E+06 \\
0.5566E+03 & 0.2968E+04 & 0.5316E+05 & 0.1284E+06 \\
0.5059E+03 & 0.9384E+04 & 0.5446E+05 & 0.1606E+06 \\
0.5190E+03 & 0.8629E+04 & 0.5716E+05 & 0.1630E+06 \\
-0.3262E+03 & -0.5416E+04 & -0.3582E+05 & -0.1022E+06
\end{array}$$

Columns 5 to 8

$$\begin{array}{cccc}
0.5566E+03 & 0.5059E+03 & 0.5190E+03 & -0.3262E+03 \\
0.2968E+04 & 0.9384E+04 & 0.8629E+04 & -0.5416E+04 \\
0.5316E+05 & 0.5446E+05 & 0.5716E+05 & -0.3582E+05 \\
0.1284E+06 & 0.1606E+06 & 0.1630E+06 & -0.1022E+06 \\
0.1446E+06 & 0.1747E+06 & 0.1782E+06 & -0.1117E+06 \\
0.1747E+06 & 0.2131E+06 & 0.2171E+06 & -0.1361E+06 \\
0.1782E+06 & 0.2171E+06 & 0.2212E+06 & -0.1387E+06 \\
-0.1117E+06 & -0.1361E+06 & -0.1387E+06 & 0.8694E+05
\end{array}$$

Our modified Cholesky gives the factorization (13) with $\kappa(\tilde{L}) = 4.716$ and $\kappa(D^2) = 1.708E+09$. Furthermore, our iterative algorithm converged after 4 sweeps during which it applied 94 rotations.

For each of the algorithms the condition of X such that $X^TBX = I$ is about $\kappa(X) = 9.E+04$.

RESIDUALS — these should not be significantly greater than 1 :

	New Algorithm	Standard Algorithm	QZ Algorithm
$\frac{\|X^TBX-I\|}{\|X\|_F^2\|B\|_F u}$	0.14	0.09	0.13
$\frac{\|X^TAX-D\|}{\|X\|_F^2\|A\|_F u}$	0.03	3.5	$5.2E+06$
$\frac{\|AXD_B-BXD_A\|}{\|X\|_F(\|A\|_F+\|B\|_F)u}$	0.30	0.25	0.53

Note that all three methods gave eigenvector matrices X which diagonalized B very effectively, and with the eigenvalue matrices D_A and D_B gave small residuals relative to the size of X. However large X can hide the fact that *some* eigenvalues could be far less accurate than necessary. Unfortunately we see the X from the QZ algorithm gives a very poor diagonalization of A, and both the standard method and our method are far superior in this respect. This "failure" of QZ for the GSEVP is quite significant in size, and surprising, as it has not been mentioned previously in the literature to our knowledge. This was confirmed by further tests performed in MATLAB with its 'qz' function, but as this presumably uses the same EISPACK routines, it did not tell us much. For the matrix B, however, our computational experience so far suggests that poor diagonalization residuals do not occur with this QZ algorithm. We have not yet looked for explanations of this behaviour.

Computed results for each method

For the new method :

Real lambda	Comp.alpha	Comp.beta	Comp.lambda	Abs.error	Sin angle
-0.6250E-01	-0.6238E-01	0.9981E+00	-0.6250E-01	0.7994E-14	0.6987E-14
0.2500E-05	0.2500E-05	0.1000E+01	0.2500E-05	0.0000E+00	0.0000E+00
0.8750E-04	0.8750E-04	0.1000E+01	0.8750E-04	0.0000E+00	0.8674E-18
0.3750E+00	0.3511E+00	0.9363E+00	0.3750E+00	0.6600E-12	0.5852E-12
0.5000E+02	0.9998E+00	0.2000E-01	0.5000E+02	0.3231E-08	0.1292E-11
0.1000E+04	0.1000E+01	0.1000E-02	0.1000E+04	0.3366E-06	0.3366E-12
0.1250E+04	0.1000E+01	0.8000E-03	0.1250E+04	0.1500E-03	0.9600E-10
0.7500E+05	0.1000E+01	0.1333E-04	0.7500E+05	0.3653E-01	0.6494E-11

For the standard method :

Real lambda	Comp.alpha	Comp.beta	Comp.lambda	Abs.error	Sin angle
-0.6250E-01	-0.6238E-01	0.9981E+00	-0.6250E-01	0.3997E-14	0.3990E-14
0.2500E-05	0.2500E-05	0.1000E+01	0.2500E-05	0.1197E-15	0.1197E-15
0.8750E-04	0.8750E-04	0.1000E+01	0.8750E-04	0.1330E-15	0.1339E-15
0.3750E+00	0.3511E+00	0.9363E+00	0.3750E+00	0.1010E-11	0.8860E-12
0.5000E+02	0.9998E+00	0.2000E-01	0.5000E+02	0.1220E-07	0.4878E-11
0.1000E+04	0.1000E+01	0.1000E-02	0.1000E+04	0.4804E-06	0.4804E-12
0.1250E+04	0.1000E+01	0.8000E-03	0.1250E+04	0.2145E-04	0.1373E-10
0.7500E+05	0.1000E+01	0.1333E-04	0.7500E+05	0.1518E-01	0.2699E-11

For the QZ method :

Real lambda	Comp.alpha	Comp.beta	Comp.lambda	Abs.error	Sin angle
-0.6250E-01	-0.6238E-01	0.9981E+00	-0.6250E-01	0.6994E-14	0.6987E-14
0.2500E-05	0.2500E-05	0.1000E+01	0.2500E-05	0.0000E+00	0.0000E+00
0.8750E-04	0.8750E-04	0.1000E+01	0.8750E-04	0.0000E+00	0.0000E+00
0.3750E+00	0.3511E+00	0.9363E+00	0.3750E+00	0.4200E-12	0.3640E-12

```
0.5000E+02  0.9998E+00  0.2000E-01  0.5000E+02  0.1795E-07  0.7177E-11
0.1000E+04  0.1000E+01  0.1000E-02  0.1000E+04  0.1026E-05  0.1026E-11
0.1250E+04  0.1000E+01  0.8000E-03  0.1250E+04  0.9434E-04  0.6038E-10
0.7500E+05  0.1000E+01  0.1333E-04  0.7500E+05  0.3938E-02  0.7000E-12
```

PERFORMANCE INDICES — again these should be compared to unity :
Computed as : $\frac{\|Ax_c\beta_c - Bx_c\alpha_c\|_2}{(|\beta_c|\|A\|_F + |\alpha_c|\|B\|_F)\|x_c\|_2 u}$

Eigenvalue	New Algorithm	Standard Algorithm	QZ Algorithm
$-0.6250E-01$	0.76	0.21	1.08
$0.2500E-05$	1.38	28766.42	0.93
$0.8750E-04$	0.28	2247.46	0.33
$0.3750E+00$	1.17	0.21	0.91
$0.5000E+02$	0.49	0.18	0.48
$0.1000E+04$	0.15	0.07	0.32
$0.1250E+04$	0.54	0.38	0.64
$0.7500E+05$	0.26	0.23	0.52

Although the standard algorithm consistently achieves a good diagonalization of both A and B, it nevertheless suffers from a lack of backward stability, as demonstrated by performance indices having large magnitudes for some eigenvalues of this test problem.

This simple but testing example suggests our algorithm can offer an improvement over the two main algorithms used for solving the GSVEP. It is reasonable to believe a more sophisticated implementation of our ideas will improve further on these results.

The *standard algorithm* used in the comparisons was that supplied in LAPACK [1], and appears to have been very well implemented, easily outperforming earlier versions we used. There was no QZ algorithm available in the LAPACK set of subroutines when we carried out these tests, but the EISPACK [4] implementation we used is well known as an effective one. That our first rough implementation of this new approach showed neither of the main weaknesses of these two suggests this new approach may be a good one.

FACTORIZATION : Below is a listing of the condition numbers of symmetric positive definite matrices together with the condition numbers for the computed $\tilde{L}$ and D^2 of the factorization (13).

$\kappa(B)$	$\kappa(\tilde{L})$	$\kappa(D^2)$
$0.2473E+03$	$0.3553E+01$	$0.4007E+02$
$0.1313E+07$	$0.6145E+01$	$0.3272E+06$
$0.1384E+09$	$0.6973E+01$	$0.1585E+08$
$0.4164E+11$	$0.1036E+02$	$0.8660E+10$
$0.4271E+13$	$0.6572E+01$	$0.6740E+12$
$0.4050E+15$	$0.3471E+01$	$0.1790E+15$

In retrospect it seems obvious that lower triangular $\tilde{L}$ with all subdiagonal elements less than unity will tend to be well-conditioned, and so D^2 will tend to contain most of the

ill-condition of B, but what is surprising is how little this has been used in problems like this.

8 Conclusion, and thoughts on Parallel Processor Implementations

We have introduced a new algorithm for the GSVEP whose computational properties appear to combine the advantages of both the standard algorithm and the QZ algorithm, two common algorithms used to solve a GSEVP, but with neither of their weaknesses. An important step of our algorithm is the use of the well-known Cholesky factorization to compute the factorization of a (possibly ill-conditioned) symmetric positive definite matrix B into its factors L and D in such a way that most of the ill-condition is reflected in D only. The resulting L was very well-conditioned in all the examples we tried, suggesting this could be used more often in similar problems involving symmetric positive definite matrices. Numerical results obtained with a preliminary FORTRAN implementation of our algorithm showed a promising improvement over the results yielded by production codes for the two other algorithms for solving the GSEVP, as well as confirming the excellent behaviour of the modified Cholesky procedure referred to above.

The pivoting requirement in the modified Cholesky factorization will limit to some extent the amount of parallelism that can be achieved in the first stage. However the Jacobi based second stage is amenable to parallel computation. This suggests that a parallel implementation could be cast in two stages:

- Solve the eigenproblem $B = QD^2Q^T$, $Q^TQ = I$ (or the SVD of G in $B = GG^T$, as this requires no pivoting) using a parallel approach, such as a Jacobi (or Kogbetliantz, see [6]), based approach;
- solve the resulting GSEVP $Q^TAQy = \lambda D^2 y$ using a parallel implementation of the implicit Jacobi method recommended here.

This would give fairly similar methods in the two stages.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide.* Philadelphia, PA, 1992.

[2] James R. Bunch. The weak and strong stability of algorithms in numerical linear algebra. *Linear Algebra and its Applications*, 88/89:49–66, 1987.

[3] C.R. Crawford. A stable generalized eigenvalue problem. *SIAM Journal on Numerical Analysis*, 13:854–860, 1976.

[4] B.S. Garbow, J.M. Boyle, J.J. Dongarra, and C.B. Moler. *Matrix Eigensystem Routines: EISPACK Guide Extension.* New York, 1977.

[5] Gene H. Golub and Charles Van Loan. *Matrix Computations (Second edition).* The Johns Hopkins University Press, Baltimore and London, 1989.

[6] M.T. Heath, A.J. Laub, C.C. Paige, and R.C. Ward. Computing the singular value decomposition of a product of two matrices. *SIAM Journal on Scientific and Statistical Computing*, 7:1147–1159, 1986.

[7] R.S. Martin and J.H. Wilkinson. Reduction of the symmetric eigenproblem $Ax = \lambda Bx$ and related problems to standard form. *Numerische Mathematik*, 11:99–110, 1968.

[8] C.B. Moler and G.W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, 10:241–256, 1973.

[9] G.W. Stewart and Ji-guang Sun. *Matrix Perturbation Theory.* Academic Press, San Diego, 1990.

[10] J.H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, London and New York, 1965.