Design of a Parallel Nonsymmetric Eigenroutine Toolbox, Part I

Zhaojun Bai* and James Demmel†

December 10, 1992

If you build it, they will come.
THE FIELD OF DREAMS

Abstract

The dense nonsymmetric eigenproblem is one of the hardest linear algebra problems to solve effectively on massively parallel machines. Rather than trying to design a "black box" eigenroutine in the spirit of EISPACK or LAPACK, we propose building a toolbox for this problem. The tools are meant to be used in different combinations on different problems and architectures. In this paper, we will describe these tools which include basic block matrix computations, the matrix sign function, 2-dimensional bisection, and spectral divide and conquer using the matrix sign function to find selected eigenvalues. We also outline how we deal with ill-conditioning and potential instability. Numerical examples are included. A future paper will discuss error analysis in detail and extensions to the generalized eigenproblem.

Appears as
- Computer Science Division Tech Report UCB/CSD-92-718, U. C. Berkeley, 1992
- Research Report 92-09, Department of Mathematics, Univ. of Kentucky, 1992.

*Department of Mathematics, University of Kentucky, Lexington, KY 40506. The author was supported in part by NSF grant ASC-9102963 and in part by DARPA grant DM28E04120 via a subcontract from Argonne National Laboratory.

†Computer Science Division and Mathematics Department, University of California, Berkeley, CA 94720. The author was supported in part by NSF grant ASC-9005933, DARPA contact DAAL03-91-C-0047 via a subcontract from the University of Tennessee, and DARPA grant DM28E04120 via a subcontract from Argonne National Laboratory.

1 Introduction

It is a challenge to design a parallel algorithm for the nonsymmetric eigenproblem that uses coarse grain parallelism effectively, scales for larger problems on larger machines, does not waste time dealing with the parts of the spectrum in which the user is not interested, and deals with highly nonnormal matrices and strongly clustered spectra. The conventional Hessenberg QR algorithm is a fine grain algorithm and has proven difficult to parallelize [5, 32, 76]. Moreover, it must find all the eigenvalues, essentially just one (or a few) at a time. In applications where only some eigenvalues are desired, one still has to compute them all. If one only wants an invariant subspace corresponding to a specified set of eigenvalues, one has to reduce the matrix completely to Schur form, and then swap the desired eigenvalues along the diagonal to group them together in order to form the desired invariant subspace [1, 4].

In this paper, we propose a collection of tools from which hybrid eigenvalue algorithms may be constructed. The resulting algorithms are easy to parallelize, and need only work on the part of the spectrum of interest to the user. The new tools use the matrix sign function to "spectrally divide and conquer" the matrix, as well as count the number of eigenvalues in a region of the complex plane. We describe how these tools might be combined to deal with different eigenvalue distributions and different user needs. We do not attempt to design a "black box" for this problem since we believe that such an algorithm would necessarily be much less efficient and reliable than one tuned for a particular application.

The algorithms we propose here are less accurate, less reliable and use more floating point operations than the best serial algorithm: Hessenberg QR iteration. On the other hand, they are easy to parallelize, and use only well understood block matrix operations like matrix multiplication, LU decomposition, and QR decomposition. In fact, all parallel algorithms that have been proposed for the nonsymmetric eigenproblem exhibit a tradeoff between parallelizability and accuracy/reliability, a tradeoff not uncommon in parallel computing [19]. Of all these algorithms, we believe our algorithm deals with this tradeoff most effectively.

Other parallel eigenalgorithms include parallel QR [5, 18, 32, 42, 69, 73, 72, 76, 77], Hessenberg divide and conquer using either Newton's method [28], or homotopies [15, 16, 28, 57, 58, 78], Jacobi's method [33, 34, 62, 65, 66, 68, 75], and reduction to nonsymmetric tridiagonal form [27, 40, 41, 43]. All these methods suffer from the use of fine-grain parallelism, instability, slow or misconvergence in the presence of clustered eigenvalues of the original problem or some constructed subproblem, or some combination of these. They must also find all the eigenvalues even if only some are desired. Our algorithm may also have difficulty converging on "difficult" parts of the spectrum, but in contrast to the above methods, this happens only if part of the original problem is very ill-conditioned *and* the user wants to compute that part of the spectrum; ill-conditioning in irrelevant parts of the spectrum need not significantly impact convergence. Furthermore, we can straightforwardly monitor the stability and convergence rate as we proceed, and either provide feedback to the user as to which part of the spectrum is difficult, or automatically compute more carefully (and slowly) to maintain stability. (See [20] for a more complete discussion of the alternate methods).

The work most closely related to ours may be found in [2, 9, 56, 61, 60]. Auslander, Bischof, Lederman and Tsao [2, 9, 56] deal with symmetric matrices, or more generally matrices with real spectra. The set of nonsymmetric matrices with all real spectra is not large, however[1]. The prior work with the matrix sign function in [60] particularly inspired us to take this approach.

We have divided this work into two papers. This first paper is organized as follows: In §2, we discuss the basic building blocks, such as the matrix sign function, which are required for higher level computations. Existing iteration and scaling schemes as well as condition estimates for the matrix sign function are also surveyed. In §3, we discuss the work of Howland [49] and Stickel [70], and show how we can use their results to count the number of eigenvalues in specified domains, or to find the corresponding invariant subspaces. This may be used as the basis either of a divide and conquer or a bisection algorithm. §4 discusses how the choice of algorithm may depend on the application. §5 outlines what we currently know about the error analysis of the algorithm; more remains to be done here. §6 presents some preliminary numerical examples and performance results. §7 discusses future research and the contents of the next paper.

2 Building Blocks

In this section, we list basic tools which serving as building blocks for solving our problem. Beside the BLAS [25, 26] and basic matrix decompositions such as LU and QR, the key tool is the matrix sign function. Different iteration and scaling schemes are collected here; since none appears uniformly superior, all are candidates for inclusion in our tool box.

2.1 Primitive Matrix Computations

1. The most primitive building blocks are the Basic Linear Algebra Subroutines, or BLAS [25, 26]. In particular matrix-matrix multiplication (`xGEMM`) will be most heavily used. We will also need the solution of triangular systems (`xTRSV` and `xTRSM`) and rank-1 matrix updates (`xGER`).

2. These BLAS are in turn used to construct routines for LU decomposition, matrix inversion, QR decomposition, condition estimation (ideally, rank revealing QR factorization [10, 13]), as well as Hessenberg reduction of a general matrix [1, 45].

It should be noted that although our mathematical focus is not on these primitive matrix computations, the performance of our algorithms will strongly depend on them. The design and performance evaluation of these basic matrix computations on shared memory machines can be found in the LAPACK Users' Guide [1] and references cited therein. The discussion of the implementation of these basic matrix computations on SIMD and MIMD machines can be found, for example, in [20, 29, 59, 60] and the references therein.

[1]Edelman [35] has recently shown that the probability that a random n by n real matrix has all real eigenvalues is very small: $2^{-n(n-1)/4}$.

2.2 Matrix Sign Function

A higher level building block is the *matrix sign function* $\text{sign}(A)$. This function was first introduced by Roberts in 1971 [63], and is defined as

$$\text{sign}(A) = 2\text{sign}^+(A) - I,$$

where the sign^+ function is defined as the contour integral

$$\text{sign}^+(A) = \frac{1}{2\pi i}\oint_{C^+} (\zeta I - A)^{-1} d\zeta.$$

Here the simple closed curve C^+ encloses all the eigenvalues of A with positive real part and no others. We assume that A has no eigenvalues with zero real part; otherwise there is a generalized matrix sign function [23]. This case will arise only as an exception in our algorithm and will be discussed later.

This expression may also be written as

$$\text{sign}(A) = P^+ - P^-$$

where

$$P^\pm = \frac{1}{2\pi i}\oint_{C^\pm} (\zeta I - A)^{-1} d\zeta.$$

Here C^- is a simple closed curve enclosing the eigenvalues of A with negative real part and no others. $P^\pm$ is the *spectral projector* for the part of the spectrum in the right (or left) halfplane [51].

We can also define the matrix sign function in the following equivalent way, as in [22]. Let

$$A = X \begin{pmatrix} J^+ & 0 \\ 0 & J^- \end{pmatrix} X^{-1}$$

be the Jordan canonical form of A, where the eigenvalues of J^+ are in open right halfplane and the eigenvalues of J^- in the open left halfplane. Then the matrix sign function of A is

$$\text{sign}(A) = X \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix} X^{-1}.$$

From this definition, it is easy to verify the following properties of $\text{sign}(A)$:

Theorem 1 . *If an $n \times n$ matrix A has no eigenvalues with zero real part, its matrix sign function* $\text{sign}(A)$ *has the following properties:*

1. $A \cdot \text{sign}(A) = \text{sign}(A) \cdot A$;
2. $(\text{sign}(A))^2 = I$ *and* $(\text{sign}(A))^{-1} = \text{sign}(A)$;
3. $\text{sign}(TAT^{-1}) = T\text{sign}(A)T^{-1}$ *for any nonsingular T;*
4. $\text{tr}(\text{sign}(A)) = \rho - \eta$ *and* $n = \rho + \eta$, *where ρ and η are the numbers of eigenvalues of A with positive and negative real parts, respectively. Equivalently,* $\rho = \frac{1}{2}(n + \text{tr}(\text{sign}(A)))$ *and* $\eta = \frac{1}{2}(n - \text{tr}(\text{sign}(A)))$.

5. *$P^{\pm} = \frac{1}{2}(I \pm \mathrm{sign}(A))$ is the spectral projector onto the eigenspace corresponding to the eigenvalues of A with positive (or negative) real parts.*

There are a variety of ways to compute the matrix sign function. The simplest iteration scheme is Newton's method applied to $(\mathrm{sign}(A))^2 = I$:

$$A_{i+1} = \frac{1}{2}(A_i + A_i^{-1}) \quad \text{with} \quad A_0 = A. \tag{2.1}$$

The iteration is globally and ultimately quadratically convergent [63, 49].

Newton iteration requires the matrix inversion A_i^{-1} which may be expensive or difficult to compute accurately. Other schemes such as Newton-Schulz iteration

$$A_{i+1} = \frac{1}{2}A_i(3I - A_i^2) \quad \text{with} \quad A_0 = A$$

require more flops (twice as many per iteration in this case) but use only matrix multiplies, which may be more efficient. The Newton-Schulz iteration is also quadratically convergent provided that $\|A_i^2 - I\| < 1$. A hybrid iteration might begin with Newton iteration until $\|A_i^2 - I\| < 1$ and then switch to Newton-Schulz iteration.

More general iteration schemes for computing the matrix sign function are based on Padé approximation [52]. In particular, the following Halley iteration:

$$A_{i+1} = A_i(3I + A_i^2)(I + 3A_i^2)^{-1}$$

is globally convergent with cubic local convergence rate. Again the tradeoff here is between the larger number of the flops per iteration versus the higher convergence rate.

There are many ways to accelerate the iterations via scaling. Two popular forms of scaling Newton's iteration are

$$A_{i+1} = \alpha_i A_i + \beta_i A_i^{-1} \quad \text{with} \quad \alpha_i + \beta_i = 1$$

and

$$A_{i+1} = \frac{1}{2}(\gamma_i A_i + \frac{1}{\gamma_i}A_i^{-1}),$$

where α_i, β_i and γ_i are appropriately chosen scalars. Some of the known scaling schemes are

Roberts [63]:	$\alpha_i = \|A_i^{-1}\|/(\|A_i\| + \|A_i^{-1}\|), \quad \beta_i = \|A_i\|/(\|A_i\| + \|A_i^{-1}\|).$
Balzer [6]:	$\alpha_i = (|\det(A_i)|^{1/n} + 1)^{-1}, \quad \beta_i = 1 - \alpha_i.$
Higham [47]:	$\gamma_i = \left[\|A_i^{-1}\|_1 \|A_i^{-1}\|_\infty / (\|A_i\|_1 \|A_i\|_\infty)\right]^{1/4}.$
Byers [12]:	$\gamma_i = |\det(A_i)|^{-1/n}.$

A comparison of different scaling schemes for different α_i and β_i is presented by Balzer [6], and for different γ_i by Kenney and Laub [54]. Kenney and Laub shows that the optimal scaling γ_i requires complete knowledge of the eigenvalues of the original matrix. A semi-optimal scaling is proposed, which only needs to know the dominant eigenvalues of the matrix and its inverse. These dominant eigenvalues may be estimated by the power method.

Note that it is convenient to calculate the determinant $\det(A_i)$ and estimate the norm $\|A^{-1}\|_\ell$, $\ell = 1$ or ∞, from the same LU or QR factors that we use to perform the matrix inversion [48]. We may also compute the matrix sign function of $p(A)$, where p is a low degree polynomial. p may be chosen so that the eigenvalue distribution of $p(A)$ has a favorable distribution for a fast convergence rate. One might also consider the matrix sign function of a matrix rational function $q(A)$, although we will not do so here.

Several stopping criteria may be used in the iterations, such as

$$\|A_{i+1} - A_i\| \le \tau\|A_i\| \quad \text{or} \quad \|A_i - A_i^{-1}\| \le \tau\|A_i\|$$

where τ is a small, user specified error bound.

As suggested by part 4 of Theorem 1 and discussed further below, we may only need to compute the trace of the sign function rounded to the nearest integer in order to use it to count the number of eigenvalues in a region of the complex plane [38, 49]. This may require a less stringent stopping criterion than for computing the entire matrix sign function.

There is a large literature on using the matrix sign function to solve algebraic matrix Riccati equations, see [55] and references therein.

2.3 Computing the sign function when A_i is ill-conditioned

When A_i is ill-conditioned, we may lose accuracy when computing $A_{i+1} = .5 \cdot (A_i + A_i^{-1})$. This may be avoided by choosing to divide the spectrum in a different place (see §3), but if the user really wants to divide the spectrum in this location, we must deal with it. To do so, we will show how to compute each iterate A_i of (2.1) to relative precision $\varepsilon_M^{1/2}$, where ε_M is the machine precision. A complete error analysis of this algorithm will be given in the second part of this two part paper.

If A has $s \ge 1$ singular values less than $\varepsilon_M^{1/2}\|A\|_2$, then there is a perturbation δA of A of norm $\|\delta A\|_2 \le \varepsilon_M^{1/2}\|A\|_2$ such that $A + \delta A$ has s zero eigenvalues. We can use a (rank revealing) QR decomposition of A to construct δA and deflate the resulting s zero eigenvalues by finding a basis for the range space of A (corresponding to the singular values exceeding $\varepsilon_M^{1/2}\|A\|_2$), and forming Q^TAQ. We set the trailing s rows of Q^TAQ to zero; this is δA. Let $\hat{A}$ denote the leading $n-s$ by $n-s$ submatrix of Q^TAQ; it is the remaining eigenproblem we must solve.

If $\hat{A}$ has condition number bounded by $\varepsilon_M^{-1/2}$, then it has no more tiny singular values less than $\varepsilon_M^{1/2}\|A\|_2$ [21, 74]. Then, using arithmetic of accuracy ε_M, we can compute $\hat{A}^{-1}$ with an absolute error bounded by

$$\varepsilon_M \kappa(\hat{A})\|\hat{A}^{-1}\|_2 \le \varepsilon_M \varepsilon_M^{-1/2}\|\hat{A}^{-1}\|_2 = \varepsilon_M^{1/2}\|\hat{A}^{-1}\|_2$$

If we using a scaling so that $\|\hat{A}\|_2$ and $\|\hat{A}^{-1}\|_2$ are not too different (either Roberts' or Higham's of the last section will do) then we can evaluate formula (2.1) with an absolute error of $O(\varepsilon_M^{1/2})\|\hat{A}\|_2$ [31]. Since the deflation introduces a backward error of size $\varepsilon_M^{1/2}$ (relative to A), the whole procedure has forward as well as backward error $\varepsilon_M^{1/2}$.

If $\hat{A}$ still has tiny singular values below $\varepsilon_M^{1/2}\|A\|_2$, then this QR deflation process could be repeated. This amounts to running the so-called *staircase algorithm* for the Jordan structure of A at zero [50]. In the worst case, if A is an n by n Jordan block with eigenvalue zero, this process will take n steps and $O(n^4)$ flops. This is, however, extremely unlikely. We discuss this complexity issue more below.

If this deflation occurs for $A_0 = A$, the deflated eigenvalues correspond to tiny eigenvalues of A as described. If it occurs for A_1, they correspond to tiny eigenvalues of A_1, or eigenvalues of A_0 near $\pm\sqrt{-1}$. Similar interpretations are possible for tiny eigenvalues of A_i for larger i. Since roundoff may make it hard to identify the corresponding eigenvalues of A_0, a very ill-conditioned A_i is probably best interpreted as a signal to split the spectrum elsewhere (see §3). For any i we can certainly construct examples where $A_0, ..., A_{i-1}$ are well-conditioned and A_i is ill-conditioned (let A_0 have a nearly pure imaginary eigenvalue λ such that iterating $\lambda \leftarrow .5(\lambda + 1/\lambda)$ i times nearly yields zero). Fortunately, this seems unlikely in practice because the iteration (2.1) moves eigenvalues near the imaginary axis away from it.

A more complete discussion of the conditioning of the matrix sign function computation will appear in part 2 of this paper. In [53, 11], some condition number estimation procedures for the matrix sign function, based on Fréchet derivatives, are discussed.

3 Higher Level Tools

We can use properties of the matrix sign function to produce two higher level tools for our toolbox:

1. counting the number of eigenvalues in a region of the complex plane, and

2. computing the invariant subspace of the eigenvalues in a region of the complex plane.

We now discuss these operations in turn, first for complex matrices, and then for real matrices. Real matrices require somewhat more complicated algorithms, because we want to avoid complex arithmetic if possible. The later numerical examples will be for real matrices only.

3.1 Counting Eigenvalues

In 1856, C. Hermite published a paper entitled "On the number of roots of an algebraic equation contained between given limits" [46]. Hermite's theory is a generalization of Sturm's theory of 1853 for the locating the eigenvalues of a symmetric matrix. The theory discusses the number of eigenvalues of a nonsymmetric matrix in a specified domain based on polynomial and associated quadratic forms. Less attention has been paid to the theory in modern times because of its apparently high computation costs and numerical sensitivity [37].

Although it is immediate to see how the matrix sign function can be used to count the number of eigenvalues in a halfplane (see part 4 of Theorem 1), Howland [49] was the first to use it to count the eigenvalues in a rectangle. The most recent work on this topic is due to Stickel [70]. The following theorems are inspired from the work of Howland and Stickel. They appear explicitly or implicitly in their work.

In the next section, we shall introduce a deflation technique for computing an invariant subspace corresponding to the eigenvalues in a specified region. This deflation technique,

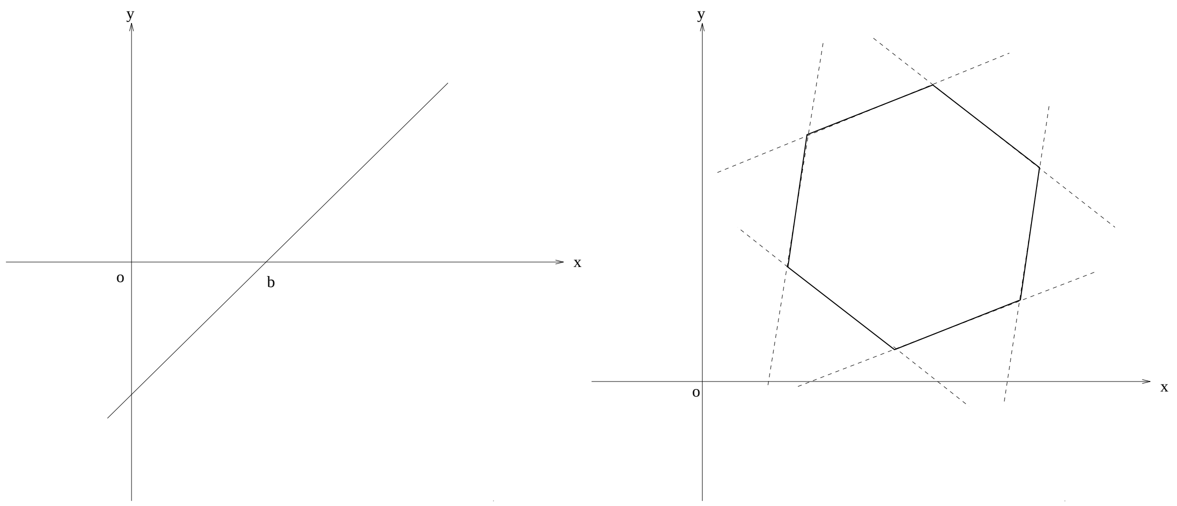

Figure 1: The half plane and convex polytope

which leads to considerable savings in practical computation, can be easily incorporated in the theorems for counting eigenvalues, although we will not do so in the interest of brevity.

First we will discuss the complex case, and then the real case.

3.1.1 Complex matrices

Part 4 of Theorem 1 tells us how to count the number of eigenvalues of A in the left or right halfplane by taking the trace of its sign function. By taking the trace of the sign function of $\alpha A + \beta I$ for appropriate scalars α and β, we can count the number of eigenvalues in any halfplane.

First, we need to define a line and its right halfplane. Let $z = \sigma e^{i\theta} + b$, σ real and $0 \le \theta < \pi$, be the locus of a line in the complex plane passing through b with slope $\tan\theta$. We may also express this locus as $\Im((z-b)e^{-i\theta}) = 0$. We will define the *right* or *positive halfplane* H_+ defined by this line as the set of z satisfying $\Im((z-b)e^{-i\theta}) > 0$, or equivalently $\Re(-i(z-b)e^{-i\theta}) > 0$. When $0 < \theta < \pi$, so the line is not horizontal, this has the natural meaning of the halfplane of points to the right of the line. When $\theta = 0$, so the line is horizontal, this is the halfplane below the boundary line. This definition is convenient because the halfplane changes continuously with θ in the range $0 \le \theta < \pi$. The *left* or *negative halfplane* H_- is defined analogously.

Theorem 2 *Let θ and b define a right halfplane H_+ as just described. Assume A has no eigenvalues on the boundary of the halfplane. Then*

$$\begin{aligned} \#\textit{ of eigenvalues in } H_+ &= \rho = \frac{1}{2}\left(\mathrm{tr}(\mathrm{sign}(-ie^{-i\theta}A + ie^{-i\theta}bI)) + n\right), \\ \#\textit{ of eigenvalues in } H_- &= n - \rho \end{aligned}$$

The theorem follows from part 4 of Theorem 1 since the eigenvalues of A in H_+ (H_-) correspond to eigenvalues of $-ie^{-i\theta}A + ie^{-i\theta}bI$ with positive (negative) real parts.

Now consider a convex polytope C in the complex plane, defined as the intersection of positive halfplanes $C = \cap_{j=1}^{k} H_+^j$, H_+^j being defined by scalars θ_j and b_j as above. The next theorem, which follows from the last one, shows how to count the number of eigenvalues of A inside C, provided C does not contain the origin (Theorem 9 will remove this restriction).

Theorem 3 *Let θ_j and b_j, $1 \le j \le k$, define a sequence of positive halfplanes H_+^j and the convex set $C \cap_{j=1}^{k} H_+^j$ as above. Assume C does not contain the origin in its boundary or interior. Assume the n-by-n matrix A has no eigenvalues on the boundaries of any H_+^j. Let $P_j = \frac{1}{2}(I + \mathrm{sign}(-ie^{-i\theta_j}A + ie^{-i\theta_j}b_jI))$ be the spectral projector onto the eigenspace of A corresponding to the eigenvalues in H_+^j (see Theorem 1, part 5). Let $\hat{A} = AP_1 \cdot P_2 \cdots P_k$. Let H_+ and H_-, defined by θ and b, satisfy $C \subset H_+$ and $0 \in H_-$. Then*

$$\#\ \textit{of eigenvalues of } A \textit{ inside } C = \frac{1}{2}(n + \mathrm{tr}(\mathrm{sign}(-ie^{-i\theta}\hat{A} + ibe^{-i\theta}))).$$

Proof. . The eigenvalues of $\hat{A}$ are the same as those of A inside C (all nonzero), and all the others are zero. This is because each multiplication by P_j leaves the eigenvalues inside H_+^j unchanged and zeros out those outside H_+^j. Therefore, the eigenvalues of $\hat{A}$ inside H_+ are identical the eigenvalues of A inside C, and the eigenvalues of $\hat{A}$ outside H_+ are all zero and equal in number to the eigenvalues of A outside C. The result now follows from part 4 of Theorem 1. ∎

Note that all the P_j in Theorem 3 commute with one another and with A, and so in principle could be computed in parallel and multiplied in any order, such as in parallel reduction.

Both these theorems require complex arithmetic in general to evaluate the sign functions, even if A is real. Since complex arithmetic is more expensive than real arithmetic, we prefer theorems that require only real arithmetic for their implementation. Such theorems are presented in the next section.

3.1.2 Real matrices

The following theorem, a special case of Theorem 2, shows that one real matrix sign function evaluation can count the number of eigenvalues of a real matrix in a halfplane with a vertical boundary.

Theorem 4 *Let $b \in \mathbb{R}$. Assume the n-by-n matrix A has no eigenvalues with real part b. Then*

$$\begin{aligned} \#\ \textit{of eigenvalues with real part greater than } b &= \rho = \frac{1}{2}\left(\mathrm{tr}(\mathrm{sign}(A - bI)) + n\right), \\ \#\ \textit{of eigenvalues with real part less than } a &= n - \rho \end{aligned}$$

With two matrix sign function evaluations, we can count the eigenvalues in a vertical strip of all complex numbers with real part in the interval (b, c).

Theorem 5 *Let $b, c \in \mathbb{R}$, $b < c$. Assume A has no eigenvalues with real part b or c. Then*

$$\#\ \textit{of eigenvalues in the strip } (b, c) = \frac{1}{2}\mathrm{tr}\left(\mathrm{sign}(A - bI) - \mathrm{sign}(A - cI)\right).$$

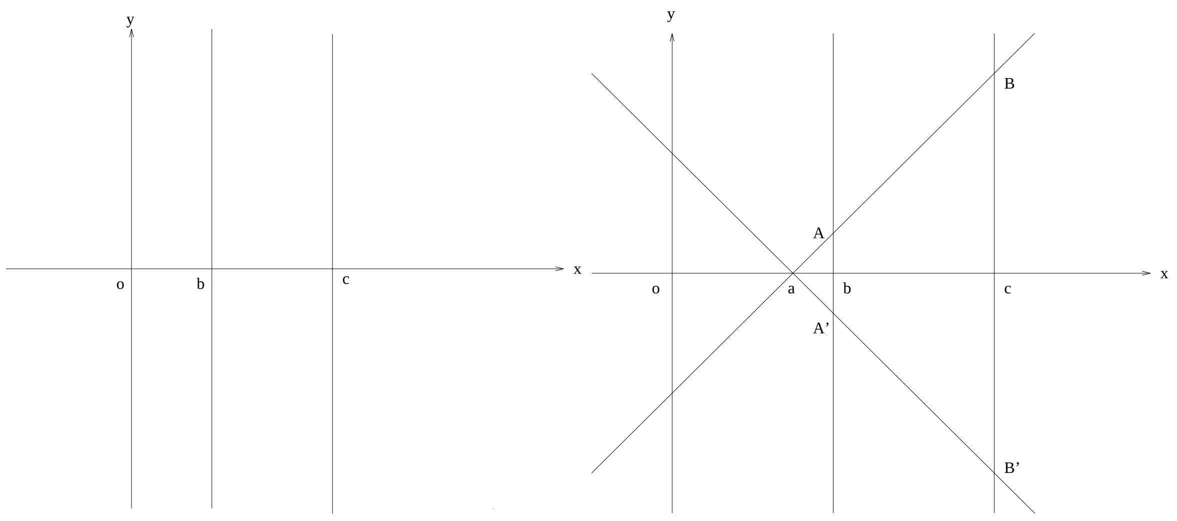

Figure 2: The vertical strip and trapezoid

With more matrix sign function evaluations, we can count the number of eigenvalues in a more complicated region. For example, three sign function evaluations can be used to count the number of eigenvalues in a trapezoid in the complex plane.

Theorem 6 *Let $a, b, c \in \mathbb{R}$, $b < c$, let the trapezoid $ABB'A'$ be defined as in Figure 2 (where $a < b < c$). Assume A has no eigenvalues on the lines containing the edges of this trapezoid. Let*

$$P_b^+ = \frac{1}{2}(I + \mathrm{sign}(A - bI)), \qquad P_c^- = \frac{1}{2}(I - \mathrm{sign}(A - cI))$$

and

$$A_p = AP_b^+ P_c^- .$$

A_p is the matrix obtained by projecting the eigenvalues of A that lie outside the vertical strip between by $x = b$ and $x = c$ onto zero while the others remain unchanged. Then

$$\#\textit{ of eigenvalues in the trapezoid } ABB'A' = \frac{1}{2}\mathrm{tr}(I + \mathrm{sign}((A_p - aI)^2)).$$

Note that the above theorem is still true if $b \le a < c$ or $c \le a$, in which cases the region bounded by lines $x = b$, $y = \pm(x - a)$ and $x = c$ may look like a triangle or butterfly.

The following theorem by Stickel [70] can be regarded as a real version of Howland's theorem [49], which shows that with four matrix sign function evaluations, one can count the number of eigenvalues in a parallelogram.

Theorem 7 *[70] Let $a, d, b, c \in \mathbb{R}$, and satisfy $a < d \le b < c$ or $b < c \le a < d$. Let the parallelogram $ABCD$ be defined as in Figure 3. Assume A has no eigenvalues on the lines containing the edges of this rectangle. Let A_p be defined as in Theorem 6. Then*

$$\#\textit{ of eigenvalues in the parallelogram } ABCD = \frac{1}{4}\mathrm{tr}\left(\mathrm{sign}((A_p - aI)^2) - \mathrm{sign}((A_p - dI)^2)\right).$$

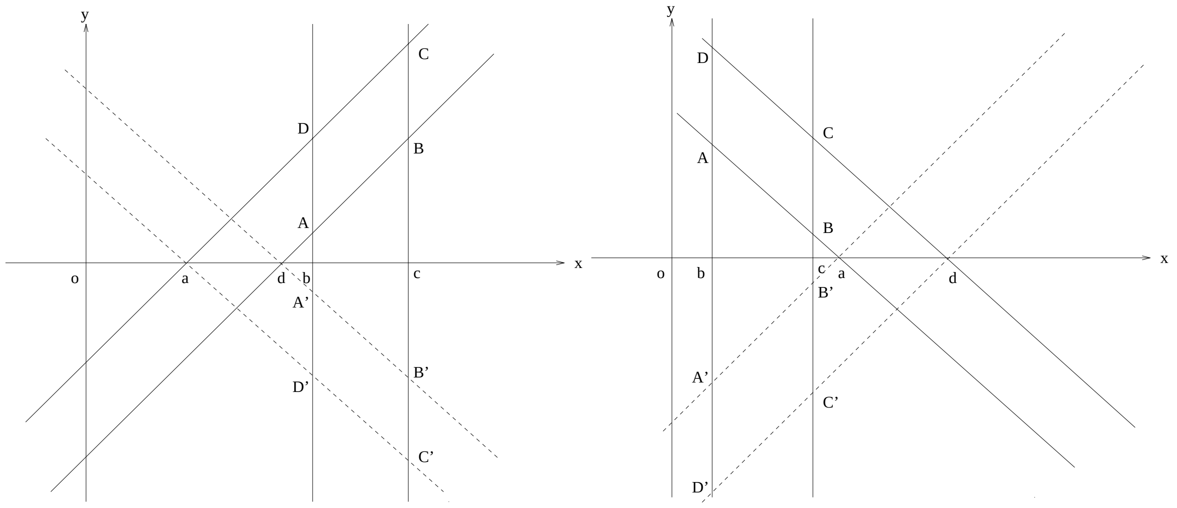

Figure 3: The parallelogram

Note that we actually know the number of eigenvalues in two parallelograms, $ABCD$ and $A'B'C'D'$, where the latter is the image of the former under reflection in the x axis.

The results of the above theorems can be used to implement one or two-dimensional bisection schemes, as shown in Figure 4. It only requires that the trace of the sign function be computed to the nearest integer. The following table counts the number of sign function (SF) evaluations for 2-D bisection if we assume that each subparallelogram in Figure 4 contains eigenvalues of A, which is a very pessimistic assumption. Column 1 in the table indicates how many levels of bisection we have performed, column 2 contains the number of new sign function evaluations at that level, column 3 contains the cumulative number of sign function evaluations, and column 4 contains the total number of (smallest) parallelograms. Note that if the deflation techniques of the next section are incorporated, we need only evaluate the matrix sign function of small matrices once we get to finer grids, which is much less expensive.

Level	# of computed SF	count of # of tr(SF)	# of parallelograms
1	4	4	1
2	5	9	4
3	16	25	16
4	56	81	64
$\vdots$	$\vdots$	$\vdots$	$\vdots$
k	$3 \cdot 2^{2(k-2)} + 2^{(k-1)}$	$2^{2(k-1)} + 2^k + 1$	$2^{2(k-1)}$

If A has eigenvalues along or near the forbidden lines, the sign function iteration will not converge (or converge very slowly), and this can be detected by condition number estimation during the Newton iteration, and the edges may be changed.

One may be able to devise schemes for counting the eigenvalues within more complicated domains, but it is not clear these would be useful in practice.

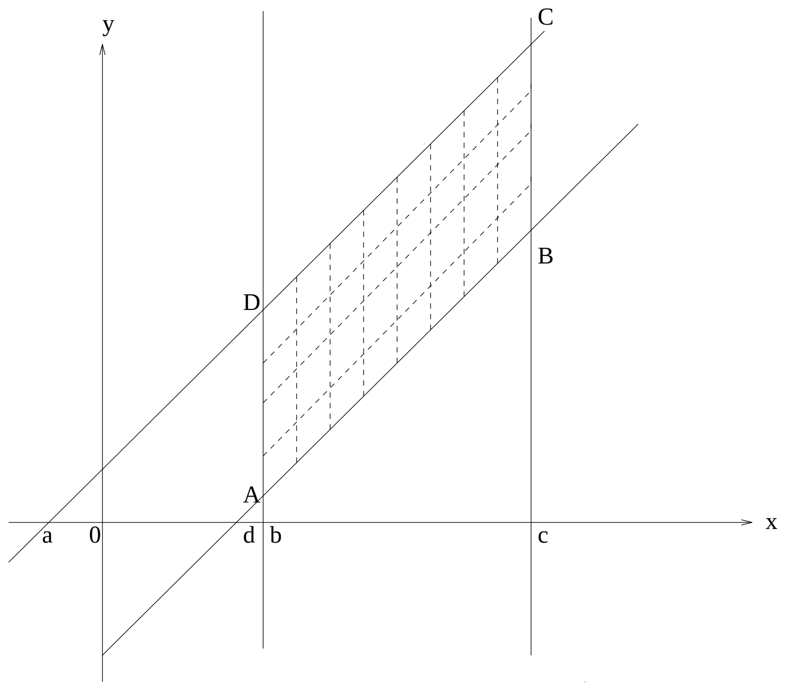

Figure 4: 2-dimensional Bisection

3.2 Computing Invariant Subspaces

We may use the matrix sign function to reduce the matrix to block upper triangular form by dividing the matrix into independent subproblems which can be solved independently and recursively. By computing a (rank revealing) QR decomposition of sign$(A) \pm I$, we can compute an orthonormal basis of the invariant subspace of the eigenvectors in the right (or left) halfplane. By computing the sign function of $\alpha A + \beta I$ as in §3.1.1, we can compute an orthonormal basis of the eigenvalues in any halfplane (although this will require complex arithmetic in general). This idea has been discussed by Beavers and Denman in 1973 [7], Denman and Leyva-Ramos in 1981 [23], and Lin and Zmijewski in 1991 [60] in the spirit of computing the complete spectral decomposition of a nonsymmetric matrix. The recent work of Auslander and Tsao in 1991 [2], and and Lederman, Tsao and Turnbull in 1992 [56] is similar, but they use a polynomial to map the eigenvalues instead of the matrix sign function; their work is limited to matrices with real eigenvalues. Theorems in this section are a summary of these results. These theorems can be considered as extensions of the theorems in the last subsection.

In order to reduce the number of floating point operations, we have developed a *deflation technique* for computing the invariant subspace corresponding to the eigenvalues in a specified region. This technique leads to considerable computational savings over the theorems in § 3.1. Throughout, we assume that no eigenvalues of A lie on the boundary lines in the complex plane determined by the sign function evaluations; we will not repeat this assumption in the statements of the theorems.

3.2.1 Complex Matrices

Theorem 8 *Let the halfplane H_+ be defined by θ and b as in Theorem 2. Let A be a matrix with no eigenvalues on the boundary of H_+. Let*

$$S = \mathrm{sign}(-ie^{-i\theta}A + ibe^{-i\theta}I) + I.$$

Let a (rank revealing) QR decomposition of S be

$$Q^T S\Pi = \begin{array}{cc} & \begin{array}{cc} k & n-k \end{array} \\ \begin{array}{c} k \\ n-k \end{array} & \begin{pmatrix} S_{11} & S_{12} \\ 0 & 0 \end{pmatrix} \end{array}$$

where Π is a permutation matrix and k is the rank of S. Then

$$Q^T A Q = \begin{array}{cc} & \begin{array}{cc} k & n-k \end{array} \\ \begin{array}{c} k \\ n-k \end{array} & \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \end{array},$$

where the k eigenvalues of A_{11} are precisely the eigenvalues of A inside H_+. The leading k columns of Q span the corresponding invariant subspace.

The above theorem reveals that with one matrix sign function evaluation, one rank revealing QR decomposition and two matrix multiplications, we can deflate the given matrix into a two by two block upper triangular matrix. The same procedure can be applied recursively and in parallel to each diagonal block until getting the desired (quasi-)triangular form. This is the main idea discussed in [60] and [2, 56]. We illustrate this below for the problem of finding the eigenvalues in a given convex polytope C and the corresponding invariant subspace.

Theorem 9 *Let the halfplanes H_+^j defined by θ_j and b_j, $1 \le j \le k$, be as in Theorem 3. The convex polytope $C = \cap_{j=1}^k H_+^j$ may be arbitrary. Then the following algorithm computes an n_k-by-n_k matrix A_k whose eigenvalues are precisely the eigenvalues of A inside C, and an n-by-n_k orthogonal matrix Q_k whose columns span the corresponding invariant subspace.*

```
$A_0 = A$;
$Q_0 = I$;
$n_0 = \dim(A)$;
for $j = 1 : k$
    Let $S_j = \mathrm{sign}(-ie^{-i\theta}A_{j-1} + ie^{-i\theta}bI)) + I$;
    Let $Q^T S_j \Pi = \begin{array}{cc} & n_j \\ n_j & \begin{pmatrix} S_{j,1} & S_{j,2} \\ 0 & 0 \end{pmatrix} \end{array}$ be a rank revealing QR decomposition of $S_j$,
        with $n_j = \mathrm{rank}(S_j)$;
    Let $A_j$ be defined by $Q^T A Q = \begin{array}{cc} & n_j \\ n_j & \begin{pmatrix} A_j & * \\ 0 & * \end{pmatrix} \end{array}$;
    $Q_j = Q_{j-1}(1 : n_0, 1 : n_{j-1}) \cdot Q(1 : n_{j-1}, 1 : n_j)$;
endfor
```

3.2.2 Real Matrices

Now we restrict ourselves to transformations requiring only real arithmetic if A is a real matrix.

Theorem 10 *Let $b \in \mathbb{R}$, and*

$$S = \frac{1}{2}(I + \operatorname{sign}(A - bI)).$$

Let a rank revealing QR decomposition of S be

$$Q^T S \Pi = \begin{array}{c c} & \begin{array}{cc} k & n-k \end{array} \\ \begin{array}{c} k \\ n-k \end{array} & \begin{pmatrix} S_{11} & S_{12} \\ 0 & 0 \end{pmatrix} \end{array}$$

where Π is a permutation matrix and k is the rank of S. Then

$$Q^T A Q = \begin{array}{c c} & \begin{array}{cc} k & n-k \end{array} \\ \begin{array}{c} k \\ n-k \end{array} & \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \end{array},$$

where the k eigenvalues of A_{11} are precisely the eigenvalues of A with real part greater than b.

In the absence of application dependent information, we can choose $b = \operatorname{tr}(A)/n$, so that neither halfplane is devoid of eigenvalues.

From Theorem 5, it is easy to see that with the evaluation of two matrix sign functions, we can deflate the invariant subspace corresponding to the eigenvalues in a vertical strip, which can be more useful. However, by using Theorem 5 directly, we need to evaluate two full $n \times n$ matrix sign functions. This work can be reduced if we use one deflation step after computing the first sign matrix, say $\operatorname{sign}(A - bI)$, because we know that the eigenvalues have been split at $x = b$ (see Theorem 10). Then the second matrix sign function is only for a submatrix with the order of the number of eigenvalues in the open right halfplane of $x = b$. This observation is stated in the following theorem.

Theorem 11 *Let $b, c \in \mathbb{R}$ and $b < c$,*

$$S_b = \frac{1}{2}(I + \operatorname{sign}(A - bI)).$$

Let a rank revealing QR decomposition of S_b be $S_b \Pi_b = Q_b R_b$, then

$$Q_b^T A Q_b = \begin{array}{c c} & \begin{array}{cc} k_b & n-k_b \end{array} \\ \begin{array}{c} k_b \\ n-k_b \end{array} & \begin{pmatrix} A_{11}^b & A_{12}^b \\ 0 & A_{22}^b \end{pmatrix} \end{array},$$

where the eigenvalues of A_{11}^b are the all eigenvalues of A in the open right halfplane of $x = b$. Define the $k_b \times k_b$ matrix

$$S_c = \frac{1}{2}(I - \operatorname{sign}(A_{11}^b - cI)).$$

Let a rank revealing QR decomposition of S_c be $S_c\Pi_c = Q_cR_c$, then

$$Q_c^T A_{11}^b Q_c = \begin{array}{c|cc} & k_c & k_b - k_c \\ \hline k_c & A_{11}^c & A_{12}^c \\ k_b - k_c & 0 & A_{22}^c \end{array}.$$

where the k_c eigenvalues of A_{11}^c are the all eigenvalues of A in the strip (b, c).

As in the case of counting the number of eigenvalues, with more (real) sign function evaluations, we can determine an invariant subspace corresponding to a more complicated region. For example, with three sign function evaluations, from Theorem 6, we can determine an invariant subspace corresponding to the eigenvalues in a trapezoid. Similarly, using deflation techniques, we can avoid computing three full $n \times n$ matrix sign functions and avoid forming the matrix A_p explicitly. This observation is stated in the following theorem.

Theorem 12 *Let $a, b, c \in \mathbb{R}$ and $b < c$. Let the trapezoid $ABB'A'$ be defined as in Figure 2 (where $a < b < c$). Let the $k_c \times k_c$ submatrix A_{11}^c be defined as in Theorem 11. Let the rank revealing QR decomposition of $S_a = \frac{1}{2}(I + \mathrm{sign}((A_{11}^c - aI)^2)))$ be $S_a\Pi_a = Q_aR_a$. Then*

$$Q_a^T A_{11}^c Q_a = \begin{array}{c|cc} & k_a & k_c - k_a \\ \hline k_a & A_{11}^a & A_{12}^a \\ k_c - k_a & 0 & A_{22}^a \end{array}.$$

where the eigenvalues of the $k_a \times k_a$ submatrix A_{11}^a are the eigenvalues of A in the trapezoid $ABB'A'$.

Note that although we still need to compute three matrix sign functions as in Theorem 6, the matrix dimensions are n, k_b and k_c, respectively, instead of n, n and n. In practice, k_b and k_c could be much smaller than n.

Theorem 7 can be used to compute an invariant subspace corresponding to the eigenvalues in the specified parallelogram. In the same spirit as Theorems 11 and 12, by incorporating the deflation technique, we may only need to compute one $n \times n$ matrix sign function, and the other matrix sign functions may be much smaller.

Theorem 13 *Let $a, d, b, c \in \mathbb{R}$, and satisfy $a < d \le b < c$ or $b < c \le a < d$. Let the k_a by k_a matrix A_{11}^a be as defined in Theorem 12. Let a rank revealing QR decomposition of $S_d = \frac{1}{2}(I - \mathrm{sign}((A_{11}^a - dI)^2)))$ be $S_d\Pi_d = Q_dR_d$, then*

$$Q_d^T A_{11}^a Q_d = \begin{array}{c|cc} & k_d & k_a - k_d \\ \hline k_d & A_{11}^d & A_{12}^d \\ k_a - k_d & 0 & A_{22}^d \end{array},$$

where the eigenvalues of the $k_d \times k_d$ submatrix A_{11}^d are the eigenvalues of A in the parallelograms $ABCD$ and $A'B'C'D'$ defined by $y = \pm(x - a), y = \pm(x - d)$ and $x = b, x = c$ as in Figure 3 (The parallelogram $A'B'C'D'$ is the reflection of $ABCD$ in the real axis).

To implement the above theorem, we need to compute sign functions of dimensions n, k_b, k_c and k_a, respectively, instead of all n's.

We end this section with a few general remarks.

(1). We can retain real arithmetic and permit more general regions than halfplanes, strips, rectangles and parallelograms by computing the sign function of (low degree) polynomials in A. For example, consider applying the sign function to $A^2 - \sigma I$, as suggested in [60]. This maps the imaginary axis to the real axis, allowing us to separate eigenvalues on or near the imaginary axis. Indeed, the locus of eigenvalues λ of A such that $\Re(\lambda^2 - \sigma) = 0$ is the hyperbola $(\Re\lambda)^2 - (\Im\lambda)^2 = \sigma$. It is obviously possible to use more general (but still low degree polynomial) maps.

(2). The QR decomposition with column pivoting (such as in LAPACK subroutine SGEQRP) can be used in the above theorems to do the rank revealing QR decomposition. This may be replaced by any more reliable or efficient rank revealing decomposition scheme.

(3). If the eigenvalues of A lie or very near the forbidden lines, the sign function iterations will converge slowly, and this can be detected.

(4). If a submatrix is small enough, we can use QR iteration, as implemented in LAPACK routine `xGEES` [1].

(5). It would be of interest to generalize the above techniques to the regular generalized eigenvalue problem $A - \lambda B$. This means that $A - \lambda B$ is square and $\det(A - \lambda B)$ is not identically zero. Gardiner and Laub [39] as well as Malyshev [61] have considered such extensions. Another possible approach, which includes the computation of both left and right deflating subspaces, has been studied. We shall report such results in the second part of this report.

4 Problem Dependent Strategies

In this section we explain why a black box based on these techniques is unlikely to be efficient in all cases, and how one could exploit application dependent information about the problem.

The most obvious application dependent optimization is to compute only those eigenvalues in regions of interest. If the user specifies the halfplane, strip, rectangle or parallelogram of interest (or more general regions formed by intersecting these or combining them with simple polynomial conformal maps), then the algorithm can begin by computing the invariant subspace for the eigenvalues in this region and deflating them. If there is difficulty in convergence, the user can be informed that there is an eigenvalue close to the boundary of the region of interest.

In addition, the user's knowledge of where eigenvalues are *unlikely* to lie can be used to choose separators whose corresponding sign functions are easy to compute.

In the absence of such information, the algorithm would have to figure out where to separate the spectrum on its own. Since one sign function evaluation may require several times as many flops as conventional Hessenberg QR, it is important to limit the number of sign function evaluations on the original matrix (later ones, on smaller matrices, are much cheaper). It is possible that an automatic "black-box" might make some bad choices in initially trying to split the spectrum and so be very expensive.

Now we consider when one may want to use the sign function to perform bisection in the complex plane. If there is an isolated tight cluster of eigenvalues, it may be adequate to know how many there are in the cluster; below we discuss why it may be expensive or ill-conditioned to ask for more information. On the other hand, a common feature of many applications is that (much of) the spectrum lies on curves in the complex plane [3]. In other words, far from forming a "cloud" or collection of small clusters, the eigenvalues lie tightly spaced on curves. In particular, there are no obvious gaps in the spectrum along which to separate it using the sign function. It is for problems like these that merely counting the numbers of eigenvalues in rectangles or parallelograms or other simple regions of the plane may be useful.

For example, suppose one knew the eigenvalues lay on curves, and suppose one can "sample" the spectrum by running inverse iteration from artfully chosen starting points. Suppose further one has sampled the spectrum finely enough to predict (via interpolation) where the spectral curves might lie. These predictions can be verified by circumscribing the predicted curves by long thin rectangles, and counting the number of eigenvalues inside. By working with the sign function of $\alpha A^2 + \beta A + \gamma$ instead of A, one can use regions whose boundaries are conic sections instead of line segments. This reduces the cost close to that of one-dimensional instead of two-dimensions bisection, which is much cheaper.

It is clear from this discussion that the running time will be "output sensitive", where "difficult" spectra, or ones for which little information is available, will take longer than easy ones. For example, consider a matrix orthogonally similar to an n by n Jordan block J with eigenvalue 0; this has extremely ill-conditioned eigenvalues. Indeed, a perturbation of norm x^n is adequate to make any $x < 1$ an exact eigenvalue. So if $n \geq 16$, for example, then any number less than .1 in magnitude is within 10^{-16} of being an exact eigenvalue. What should the output of the routine be in this case? The result from Hessenberg QR algorithm to which we are accustomed is that all n computed eigenvalues are the exact eigenvalues for a matrix differing from the input by little more than machine precision (10^{-16}, say). So we might anticipate getting most eigenvalues lying evenly spaced on a circle of radius $(10^{-16})^{1/n}$, summing to a value no larger than about 10^{-16}. How does the sign function iteration behave on this matrix? J's condition number is at least about 10^{16}, so we can not expect much accuracy from the inverse. If we shift J by any number $|x| < 1$, then the condition number is still at least about $|x|^{-n}$. If we use QR decomposition to deflate out the null space, we discover the null space is one dimensional, and the remaining $n-1$ by $n-1$ matrix is similar to an $n-1$ by $n-1$ Jordan block and just as difficult to handle numerically. If we continue deflating until the remaining submatrix is moderately well-conditioned, we will have to deflate off nearly every eigenvalue serially, at a cost of $O(n^4)$ flops. This is the staircase algorithm.[2]

In the language of [71], nearly the entire unit disk is the "pseudospectrum" of a large Jordan block, since a tiny perturbation of the matrix can make any complex number in the disk an exact eigenvalue. This assumes any perturbation of sufficiently small *norm* is permitted. Of course the user's data may have more structure than that, and so the spectrum may be much better defined. For example, a componentwise relative approach to eigenvalue uncertainty has been considered [1, 44, 14]. Unfortunately, computing eigenvalues this accu-

[2]A worst case $O(n^3)$ variation of the staircase algorithm exists, but it seems unsuited to coarse grain parallelism [8].

Subroutine	Parameters	Region	Cutting lines
xHALFP	b	halfplane	$x = b$
xSTRIP	b, c	vertical strip	$x = b, c$
xTRAPE	$a, b, c^{\dagger}$	trapezoid or butterfly	$y = \pm(x-a), x = b, c,$
xPLLGM	$a, d, b, c^{\ddagger}$		
	if $d \leq b$ or $c \leq a$	parallelogram	$y = \pm(x-a), \pm(x-d), x = b, c$
	if $a \leq b < d$	trapezoid	$y = \pm(x-a), x = b, c$
	if $b < a < d \leq c$	butterfly	$x = b, c, y = \pm(x-a)$
	if $b < a < c < d$	trapezoid	$x = b, c, y = \pm(x-d)$

x = 'S': single precision, x = 'D': double precision

$\dagger : b < c$, and $\ddagger : a < d, b < c$.

Table 1: Subroutines and their corresponding regions, cutting lines

rately is not easy even using sophisticated serial algorithms. Existing parallel nonsymmetric eigenproblem algorithms [20, 28, 57] are not guaranteed to attain the conventional normwise level of stability.

5 Numerical Experiments

In this section, we will discuss the numerical aspects of the tools described above as well as their performance. A set of FORTRAN 77 subroutines has been written to implement the different schemes for computing eigenvalues of a real nonsymmetric matrix in a specified region. Table 1 is a list of subroutine names, parameters required to determine the desired regions and their cutting lines. These subroutines are built on the top of the BLAS and LAPACK subroutines for the basic matrix operations, LU decomposition, QR decomposition and so on. The algorithms of these subroutines are described in Theorems 10, 11, 12 and 13. Each subroutine computes an orthogonal matrix Q, such that

$$Q^T A Q = \begin{array}{c c} & \begin{array}{cc} k & n-k \end{array} \\ \begin{array}{c} k \\ n-k \end{array} & \begin{pmatrix} A_{11} & A_{12} \\ E_{21} & A_{22} \end{pmatrix} \end{array}, \tag{5.2}$$

where the eigenvalues of the k by k matrix A_{11} are the eigenvalues of A in the desired region, and $\|E_{21}\|$ reveals the accuracy of computed invariant subspace. In exact arithmetic, E_{21} is zero.

To test the numerical behavior of the tools, we let the $n \times n$ real nonsymmetric test matrices have Schur decomposition

$$A = Q^T T Q$$

where Q is a random orthogonal matrix and T is a upper quasi triangular matrix (block upper triangular with 1-by-1 and 2-by-2 diagonal blocks). The 2-by-2 diagonal blocks of T

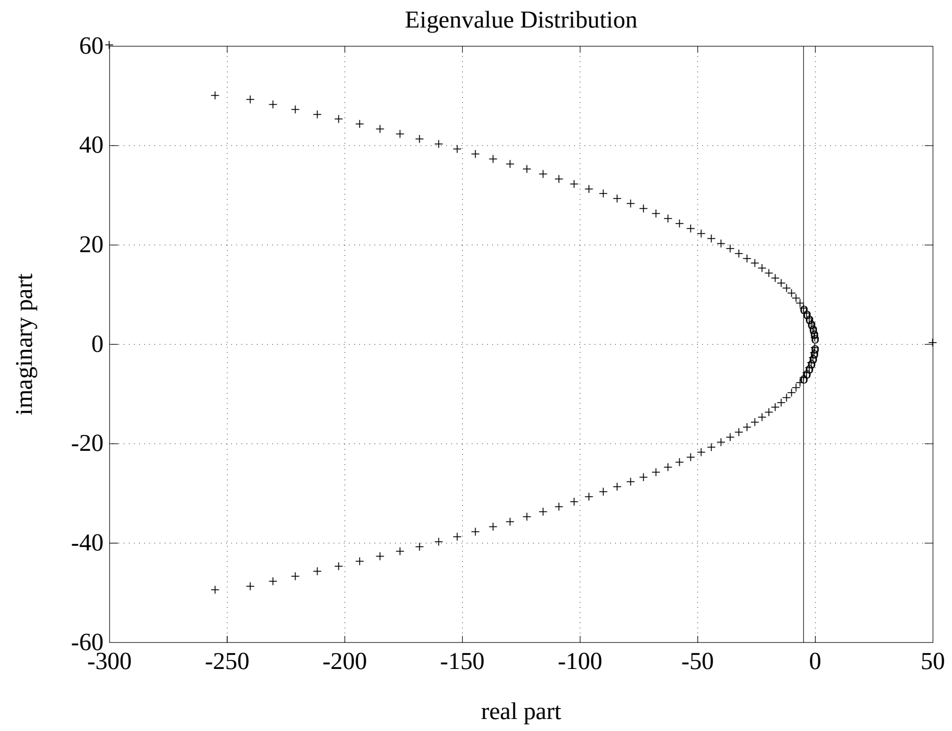

Figure 5: The eigenvalue distribution of the matrix A in Example 1

have the form

$$\begin{pmatrix} x_k & y_k \\ -y_k & x_k \end{pmatrix}$$

so their eigenvalues are $x_k \pm iy_k$. The offdiagonal part of T can be chosen to control the departure from normality of matrix A and the condition number $\kappa(A) = \|A\| \cdot \|A^{-1}\|$.

The tests were carried out on a SUN workstation 1^+ using IEEE standard double precision arithmetic, with machine epsilon $= \varepsilon_M = 2^{-52} \approx 2.2 \times 10^{-16}$.

Example 1. The first example is a matrix suggested by Françoise Chatelin, which simulates the matrices arising in aerodynamic stability analysis. The eigenvalues in this application lie on certain curves. The most interesting eigenvalues in stability analysis are those eigenvalues closest to the imaginary axis. We need to know whether there are eigenvalues with positive real part, and otherwise how close they are to the imaginary axis.

In this experiment, we choose $n = 100$, $x_k = -k^2/10$, $y_k = -k$, $k = \pm 1, \pm 2, \ldots, \pm n/2$, i.e., the eigenvalues lie on the parabola $x = -y^2/10$. The condition number of the test matrix is $\kappa(A) \approx 1.12 \times 10^6$. We use the simplest Newton iteration (2.1) for computing the matrix sign function without any scaling. The stopping criterion for Newton iteration is

$$\frac{\|A_{j+1} - A_j\|_1}{\|A_j\|_1} \leq n\epsilon_M \approx 2.2 \times 10^{-14}. \tag{5.3}$$

The computational tasks are to find those eigenvalues closest to the pure imaginary axis and their corresponding invariant subspace. Therefore, using the matrix sign function, we only compute the eigenvalues in the open right halfplane of $x = -5$.

For the computed invariant subspace corresponding to the open right halfplane $x < -5$, we have

$$\bar{Q}^T A \bar{Q} = \begin{array}{c c} & \begin{array}{cc} 14 & 86 \end{array} \\ \begin{array}{c} 14 \\ 86 \end{array} & \begin{pmatrix} A_{11} & A_{12} \\ E_{21} & A_{22} \end{pmatrix} \end{array},$$

with $\|E_{21}\|_1 = 1.70 \times 10^{-11}$, and the eigenvalues of A_{11} are the 14 eigenvalues of A in the open right halfplane $x > -5$. By comparing the computed eigenvalues and the exact eigenvalues in this region, they agree to 11 decimal digits. The computed eigenvalues closest to imaginary axis are

```
-9.999999999982578e-02 + 1.000000000000703e+00i
-9.999999999982578e-02 - 1.000000000000703e+00i
```

and the known exact ones are $-0.1 \pm i$.

It took 14 unscaled Newton iterations to converge to the desired accuracy for the matrix sign function. Overall, it requires about $31n^3$ flops to determine these eigenvalues and the corresponding invariant subspace. Note that it is competitive with the cost of QR algorithm. The QR algorithm generally takes about $22n^3$ flops to compute the Schur decomposition (both eigenvalues and orthogonal transformation matrix) plus the cost of swapping the selected eigenvalues together, which also costs $O(n^3)$.

Example 2. In this example, we construct the test matrix with half of the eigenvalues real, and the other half complex conjugates lying on a parabola. This example simulates certain bifurcation phenomena, in which the eigenvalues depend on a parameter. The eigenvalues of the matrix could change from real to complex with the change of parameter. When such a change happens, we say bifurcation occurs. We are interested to know whether the bifurcation happens in a specified region. Thus, we only need to find the eigenvalues in the specified region.

In our experiment, we let $n = 80$, and the eigenvalue distribution is shown in Figure 6. The condition number of the matrix is $\kappa(A) \approx 4.7 \times 10^4$. We used the same unscaled Newton iteration and stopping criterion as in Example 1. We computed the eigenvalues (and their corresponding invariant subspace) in a vertical strip defined by the lines $x = \pm 5$.

Using Theorem 8, for the computed invariant subspace corresponding to the eigenvalues in the vertical strip $(-5, 5)$, we have

$$\bar{Q}^T A \bar{Q} = \begin{array}{c c} & \begin{array}{cc} 16 & 64 \end{array} \\ \begin{array}{c} 16 \\ 64 \end{array} & \begin{pmatrix} A_{11} & A_{12} \\ E_{21} & A_{22} \end{pmatrix} \end{array}$$

with $\|E_{21}\|_1 = 4.09 \times 10^{-12}$. The eigenvalues of A_{11} are the 16 eigenvalues of A in the strip.

It took 12 Newton iterations to determine 42 eigenvalues of A at the open right halfplane of $x = -5$, then for the 42 by 42 submatrix A'_{11}, it took 14 Newton iterations to determine 16 eigenvalues of A'_{11} in the strip. There are 2 real eigenvalues, and 14 complex conjugate eigenvalues in the strip. All computed eigenvalues are correct to 12 decimal digits.

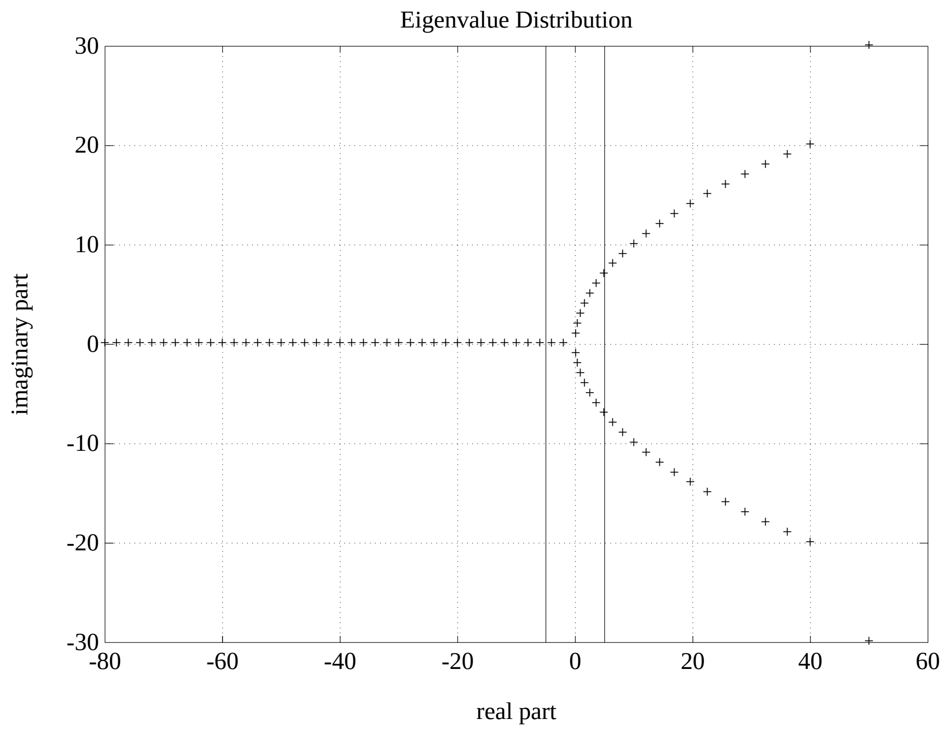

Figure 6: The eigenvalue distribution of the matrix A in Example 2

Concerning the effects of different scaling schemes in the Newton iteration with the stopping criterion (5.3), we tried all scaling schemes mentioned in §2 for the above two test matrices. The following table shows the number of iterations needed to converge, where K & L means Kenney and Laub's semi-optimal scheme. The norm used in Roberts and Balzer's schemes is the 1-norm. The last column in the table is for Hally iteration. Although Hally iteration took fewer iterations, it costs significantly more flops than Newton iteration. Kenney and Laub's semi-optimal scaling scheme took fewer iteration steps, however, it needs to do complex arithmetic in the power method. Therefore, the actual execution time turned out to be a little longer than some of the other scaling schemes.

matrix	unscaled	Byers	Higham	Roberts	Balzer80	K & L	Halley
Ex. 1	14	14	13	13	11	11	9
Ex. 2	12	13	16	15	11	11	8

Performance: We report preliminary timing results. The timing of computing the eigenvalues in a specified region using matrix sign function tool and standard QR algorithm plus swapping was performanced on a CRAY Y-MP (one processor) with 6.0 nsec clock, and $\varepsilon_M = 1.4210 \times 10^{-14}$.

The test matrices were chosen as random matrices with normally distributed entries (mean 0, variance 1). The matrix sign functions were computed by Newton iteration with

	Half plane			Strip			Parallelogram		
matrix	CPU time		**Speedup**	CPU time		**Speedup**	CPU time		**Speedup**
order	Sign	QR	QR/Sign	Sign	QR	QR/Sign	Sign	QR	QR/sign
50	0.0695	0.0958	**1.38**	0.072	0.104	**1.44**	0.088	0.0858	**0.98**
100	0.266	0.440	**1.65**	0.363	0.443	**1.22**	0.270	0.372	**1.38**
200	1.71	2.39	**1.40**	1.56	2.27	**1.46**	1.69	2.11	**1.25**
300	4.33	6.52	**1.51**	4.51	6.35	**1.41**	4.34	5.77	**1.33**
400	9.07	14.5	**1.60**	10.2	13.6	**1.33**	8.98	12.4	**1.38**

Table 2: Timing in seconds and speedups, CRAY Y-MP, 6.0 nsec clock, 1 processor, UNICOS 6.1, CFT77 5.0, libsci BLAS

Byers' determinantal scaling scheme (see §2.2). The stopping criterion was

$$\frac{\|A_{j+1} - A_j\|_1}{\|A_j\|_1} \leq n\epsilon_M \ .$$

Table 2 compares the timing of subroutines `SHALFP`, `SSTRIP` and `SPLLGM` for computing the eigenvalues in open right halfplane of $x = 0$, in the vertical strip $(-2, 2)$ and in the parallelograms defined by the lines $y = \pm(x + 2)$, $y = \pm x$ and $x = 0$, $x = 4$. Besides the timing results reported in Table 2, we also computed the number of eigenvalues in these regions counted by the subroutines `SHALFP`, `SSTRIP` and `SPLLGM` and compared to the QR algorithm with swapping (LAPACK driver routine `SGEES`); they all agree. The accuracy of the computed invariant subspaces are all of order ${\varepsilon_M}^{2/3}$, i.e., $\|E_{21}\|_1 = O({\varepsilon_M}^{2/3})$ in (5.2).

Table 2 shows that although the matrix sign function subroutines `SSTRIP` and `SPLLGM` require about 50% to 70% more floating point operations than the standard QR algorithm plus swapping, they are usually about 1.3 to 1.5 times faster. This is because most floating point operations in the sign function are large block matrix operations like matrix-matrix multiplication, whereas QR iteration performs only small matrix-vector operations. (About 80% of the flops in the sign function algorithm are spent computing the LU decomposition and matrix inverse). We expect this advantage to be magnified on massively parallel machines.

6 Future Work

This section presents a step-by-step outline of software that needs to be developed, test matrices that need to be run, and comparisons that need to be made to evaluate the proposed algorithm.

Basic building blocks: The BLAS as well as higher level building blocks like QR and LU decompositions and condition estimators exist on high performance workstations and shared memory machines [1], and work is underway to provide them on distributed memory machines [30]. A rank-revealing factorization and staircase algorithm still need to be written. In the spirit of using the fastest (if less reliable) building blocks, we could use LU plus condition estimation for our rank detection, since this will work most of the time.

Our tool box should certainly contain more sophisticated QR based algorithms in case LU fails.

Matrix sign function: The numerical computation of the sign function needs to be further developed. It should have several options available, including different iterations, scalings, stopping criteria, and a way to detect and deal with very ill-conditioned A_i. The implementation of matrix sign function computation on distributed memory machines will follow once the basic building blocks are available on these machines.

Error Analysis: We have a partial understanding of when we expect the sign function to converge, and how accurately we can compute it. Our error bounds are for the worst case, and often pessimistic. We need to devise better a posteriori error estimates for use as computational diagnostics.

Extension to the generalized eigenproblem $A - \lambda B$: An analog of the Newton iteration (2.1) for $A - \lambda B$ is $A_{i+1} - \lambda B = \frac{1}{2}(A_i + BA_i^{-1}B) - \lambda B$; this converge to an A_∞ such that $A_\infty B^{-1}$ is the matrix sign function of AB^{-1}. Based on this observation, we can try to extend our techniques to the generalized eigenproblem.

Inverse iteration and iterative refinement: Inverse iteration can be used to find an eigenvalue close to a given starting value (but not necessarily the nearest one), to sample the spectrum in selected locations and to steer a bisection based algorithm as described in §4. It may also be used to refine an approximate invariant subspace computed by other means, such as nonsymmetric Jacobi-Schur method.

User interface: A rudimentary interactive graphics based driver has been developed [24]. It lets the user select a region of the complex plane for which to compute the sign function, and then refine it. It needs to be augmented to maintain a data structure containing information about the regions of the plane which have been searched, deflated, or perhaps completely solved using QR algorithm; how many eigenvalues are in each region; how many iterations and how much CPU time was required for each sign function iteration; and which diagonal subblocks of the transformed matrix correspond to each region.

Numerical tests: The test cases in [3] as well as others of practical interest should be subjected to this toolbox. We should compare running times with competitive codes (QR algorithm from LAPACK, and Arnoldi and Lanczos methods [64, 67, 17, 36]). We can evaluate the usefulness of problem dependent knowledge by running examples with users who either know nothing about the matrix except perhaps Gershgorin information, or are given some rough information about the spectrum.

Acknowledgement

The authors would like to acknowledge R. Byers, N. Higham and A. Laub for fruitful discussion on this work and his invaluable comments on the manuscript. We would like to acknowledge E. Anderson for help in performing numerical experiments on CRAY Y-MP computers. This work was partially performed during a visit to the Institute for Mathematics and its Applications at the University of Minnesota.

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, Release 1.0.* SIAM, Philadelphia, 1992.

[2] L. Auslander and A. Tsao. On parallelizable eigensolvers. *Advances in Applied Mathematics*, 13:253–261, 1992.

[3] Z. Bai. A collection of test matrices for the large sparse nonsymmetric eigenvalue problem. in preparation.

[4] Z. Bai and J. Demmel. On swapping diagonal block in real Schur form. to appear in Lin. Alg. Appl.

[5] Z. Bai and J. Demmel. On a block implementation of Hessenberg multishift QR iteration. *International Journal of High Speed Computing*, 1(1):97–112, 1989. (also LAPACK Working Note #8).

[6] L. A. Balzer. Accelerated convergence of the matrix sign function method of solving Lyapunov, Riccati and other matrix equations. *Inter. J. Control*, 32:1057–1078, 1980.

[7] A. N. Beavers Jr and E. D. Denman. A computational method for eigenvalue and eigenvectors of a matrix with real eigenvalues. *Numer. Math.*, 21:389–396, 1973.

[8] T. Beelen and P. Van Dooren. An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 105:9–65, 1988.

[9] C. Bischof and X. Sun. A divide and conquer method for tridiagonalizing symmetric matrices with repeated eigenvalues. MCS Report P286-0192, Argonne National Lab, 1992.

[10] C. Bischof and P. Tang. Robust incremental condition estimation. Computer Science Dept. Technical Report CS-91-133, University of Tennessee, Knoxville, 1991. (LAPACK Working Note #33).

[11] R. Byers. Numerical stability and instability in matrix sign function based algorithms. In C. Byrnes and A. Lindquist, editors, *Computational and Combinatorial Methods in Systems Theory*, pages 185–200. North-Holland, 1986.

[12] R. Byers. Solving the algebraic Riccati equation with the matrix sign function. *Lin. Alg. Appl.*, 85:267–279, 1987.

[13] T. Chan. Rank revealing QR factorizations. *Lin. Alg. Appl.*, 88/89:67–82, 1987.

[14] F. Chatelin. *Valeurs propres de matrices.* Masson, Paris, 1988. English translation to appear (Wiley).

[15] M. Chu. A note on the homotopy method for linear algebraic eigenvalue problems. *Lin. Alg. Appl*, 105:225–236, 1988.

[16] M. Chu, T.-Y. Li, and T. Sauer. Homotopy method for general λ-matrix problems. *SIAM J. Mat. Anal. Appl.*, 9(4):528–536, 1988.

[17] J. Cullum and R. A. Willoughby. A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices. In J. Cullum and R. A. Willoughby, editors, *Large Scale Eigenvalue Problems*. North-Holland, Amsterdam, 1986.

[18] G. Davis, R. Funderlic, and G. Geist. A hypercube implementation of the implicit double shift QR algorithm. In *Hypercube Multiprocessors 1987*, pages 619–626, Philadelphia, PA, 1987. SIAM.

[19] J. Demmel. Trading off parallelism and numerical stability. Computer Science Division Tech Report UCB//CSD-92-702, University of California, Berkeley, CA, 1992. to appear in NATO-ASI Proceedings, 1992.

[20] J. Demmel, M. Heath, and H. van der Vorst. Parallel numerical linear algebra. In A. Iserles, editor, *Acta Numerica, volume 2*. Cambridge University Press, 1993 (to appear).

[21] J. Demmel and B. Kågström. Stably computing the kronecker structure and reducing subspaces of singular pencils $A - \lambda B$ for uncertain data. In J. Cullum and R. A. Willoughby, editors, *Large Scale Eigenvalue Problems*. North-Holland, Amsterdam, 1986.

[22] E. D. Denman and A. N. Beavers Jr. The matrix sign function and computations in systems. *Appl. Math. Comput.*, 2:63–94, 1976.

[23] E. D. Denman and J. Leyva-Ramos. Spectral decomposition of a matrix using the generalized sign matrix. *Appl. Math. Comput.*, 8:237–250, 1981.

[24] J. Dongarra. private communication. 1992.

[25] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.

[26] J. Dongarra, J. Du Croz, S. Hammarling, and Richard J. Hanson. An extended set of fortran basic linear algebra subroutines. *ACM Trans. Math. Soft.*, 14(1):1–17, March 1988.

[27] J. Dongarra, G. A. Geist, and C. Romine. Computing the eigenvalues and eigenvectors of a general matrix by reduction to tridiagonal form. Technical Report ORNL/TM-11669, Oak Ridge National Laboratory, 1990. to appear in *ACM TOMS*.

[28] J. Dongarra and M. Sidani. A parallel algorithm for the non-symmetric eigenvalue problem. Computer Science Dept. Technical Report CS-91-137, University of Tennessee, Knoxville, TN, 1991.

[29] J. Dongarra and R. van de Geijn. Reduction to condensed form for the eigenvalue problem on distributed memory computers. Computer Science Dept. Technical Report CS-91-130, University of Tennessee, Knoxville, 1991. (LAPACK Working Note #30), to appear in *Parallel Computing*.

[30] J. Dongarra, R. van de Geijn, and D. Walker. Look at scalable dense linear algebra libraries. In *Scalable High-Performance Computing Conference.* IEEE Computer Society Press, April 1992.

[31] J. Du Croz and N. J. Higham. Stability of methods for matrix inversion. *IMA J. Num. Anal.*, Jan 1992. (LAPACK Working Note #27).

[32] A. Dubrulle. The multishift QR algorithm: is it worth the trouble? Palo Alto Scientific Center Report G320-3558x, IBM Corp., 1530 Page Mill Road, Palo Alto, CA 94304, 1991.

[33] P. Eberlein. A Jacobi method for the automatic computation of eigenvalues and eigenvectors of an arbitrary matrix. *J. SIAM*, 10:74–88, 1962.

[34] P. Eberlein. On the Schur decomposition of a matrix for parallel computation. *IEEE Trans. Comput.*, 36:167–174, 1987.

[35] A. Edelman. On the distribution of a scaled condition number. *Math. Comp.*, 58(197):185–190, 1992.

[36] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. Technical Report 91-05, IPS ETH, Zürich, 1991.

[37] F. Gantmacher. *The Theory of Matrices, vol. II (transl.).* Chelsea, New York, 1959.

[38] I. Garantini and P. Henrici. Circular arithmetic and the determination of polynomial zeros. *Numer. Math.*, 18:305–320, 1972.

[39] J.D. Gardiner and A.J. Laub. A generalization of the matrix-sign function solution for algebraic riccati equations. *Int. J. Control*, 44:823–832, 1986.

[40] G. A. Geist. Parallel tridiagonalization of a general matrix using distributed memory multiprocessors. In *Proceedings of the Fourth SIAM Conference on Parallel Processing for Scientific Computing*, pages 29–35, Philadelphia, PA, 1990. SIAM.

[41] G. A. Geist. Reduction of a general matrix to tridiagonal form. *SIAM J. Mat. Anal. Appl.*, 12(2):362–373, 1991.

[42] G. A. Geist and G. J. Davis. Finding eigenvalues and eigenvectors of unsymmetric matrices using a distributed memory multiprocessor. *Parallel Computing*, 13(2):199–209, 1990.

[43] G. A. Geist, A. Lu, and E. Wachspress. Stabilized reduction of an arbitrary matrix to tridiagonal form. Technical Report ORNL/TM-11089, Oak Ridge National Laboratory, 1989.

[44] A. J. Geurts. A contribution to the theory of condition. *Num. Math.*, 39:85–96, 1982.

[45] G. Golub and C. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, 2nd edition, 1989.

[46] C. Hermite. On the number of roots of an algebraic equation contained between given limits. *J. Reine Angew. Math.*, 52:39–51, 1856. English translation, P. C. Parks, Internat. J. Control, 26:183–195,1977.

[47] N. J. Higham. Computing the polar decomposition - with application. *SIAM J. Sci. Stat. Comput.*, 7:1160–1174, 1986.

[48] N. J. Higham. A survey of condition number estimation for triangular matrices. *SIAM Review*, 29:575–596, 1987.

[49] J. Howland. The sign matrix and the separation of matrix eigenvalues. *Lin. Alg. Appl.*, 49:221–232, 1983.

[50] B. Kågström and A. Ruhe. An algorithm for the numerical computation of the Jordan normal form of a complex matrix. *ACM Trans. Math. Soft.*, 6(3):389–419, 1980.

[51] T. Kato. *Perturbation Theory for Linear Operators.* Springer Verlag, Berlin, 2 edition, 1980.

[52] C. Kenney and A. Laub. Rational iteration methods for the matrix sign function. *SIAM J. Mat. Anal. Appl.*, 21:487–494, 1991.

[53] C. Kenney and A. J. Laub. Polar decomposition and matrix sign function condition estimates. *SIAM J. Sci. Stat. Comput.*, 12:488–504, 1991.

[54] C. Kenney and A. J. Laub. On scaling Newton's method for polar decomposition and the matrix sign function. *SIAM J. Mat. Anal. Appl.*, 13(3):688–706, 1992.

[55] A. J. Laub. Invariant subspace method for the numerical solution of Riccati equations. In A. J. Laub S. Bittanti and J. C. Willems, editors, *Riccati Equations*, Berlin, June 4-6 1990. Springer-Verlag.

[56] S. Lederman, A. Tsao, and T. Turnbull. A parallelizable eigensolver for real diagonalizable matrices with real eigenvalues. Report TR-01-042, Supercomputing Research Center, Bowie, MD, 1992.

[57] T.-Y. Li and Z. Zeng. Homotopy-determinant algorithm for solving nonsymmetric eigenvalue problems. *Math. Comp.*, 59(200):483–502, 1992.

[58] T.-Y. Li, Z. Zeng, and L. Cong. Solving eigenvalue problems of nonsymmetric matrices with real homotopies. *SIAM J. Num. Anal.*, 29(1):229–248, 1992.

[59] W. Lichtenstein and S. L. Johnsson. Block cyclic dense linear algebra. Technical report, Thinking Machines Corporation, Cambridge, MA, 1992.

[60] C-C. Lin and E. Zmijewski. A parallel algorithm for computing the eigenvalues of an unsymmetric matrix on an SIMD mesh of processors. Department of Computer Science TRCS 91-15, University of California, Santa Barbara, CA, July 1991.

[61] A. N. Malyshev. Parallel aspects of some spectral problems in linear algebra. Dept. of Numerical Mathematics Report NM-R9113, Centre for Mathematics and Computer Science, Amsterdam, July 1991.

[62] M.H.C. Paardekooper. A quadratically convergent parallel Jacobi process for diagonally dominant matrices with distinct eigenvalues. *J. Comput. Appl. Math.*, 27:3–16, 1989.

[63] J. Roberts. Linear model reduction and solution of the algebraic Riccati equation. *Inter. J. Control*, 32:677–687, 1980.

[64] Y. Saad. Numerical solution of large nonsymmetric eigenvalue problems. *Comput. Phys. Comm.*, 53:71–90, 1989.

[65] A. Sameh. On Jacobi and Jacobi-like algorithms for a parallel computer. *Math. Comp.*, 25:579–590, 1971.

[66] G. Shroff. A parallel algorithm for the eigenvalues and eigenvectors of a general complex matrix. *Num. Math.*, 58:779–805, 1991.

[67] D. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Mat. Anal. Appl.*, 13(1):357–385, 1992.

[68] G. W. Stewart. A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian matrix. *SIAM J. Sci. Stat. Comput.*, 6:853–864, 1985.

[69] G. W. Stewart. A parallel implementation of the QR algorithm. *Parallel Computing*, 5:187–196, 1987.

[70] E. Stickel. Separating eigenvalues using the matrix sign function. *Lin. Alg. Appl.*, 148:75–88, 1991.

[71] L. N. Trefethen. Non-normal matrices and pseudo-eigenvalues. book in preparation.

[72] R. van de Geijn. *Implementing the QR Algorithm on an Array of Processors*. PhD thesis, University of Maryland, College Park, August 1987. Computer Science Department Report TR-1897.

[73] R. van de Geijn and D. Hudson. Efficient parallel implementation of the nonsymmetric QR algorithm. In J. Gustafson, editor, *Hypercube Concurrent Computers and Applications*. ACM, 1989.

[74] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Lin. Alg. Appl.*, 27:103–141, 1979.

[75] K. Veselić. A quadratically convergent Jacobi-like method for real matrices with complex conjugate eigenvalues. *Num. Math.*, 33:425–435, 1979.

[76] D. Watkins. Shifting strategies for the parallel QR algorithm. Dept. of pure and applied math. report, Washington State Univ., Pullman, WA, 1992.

[77] D. Watkins and L. Elsner. Convergence of algorithms of decomposition type for the eigenvalue problem. *Lin. Alg. Appl.*, 143:19–47, 1991.

[78] Z. Zeng. *Homotopy-determinant algorithm for solving matrix eigenvalue problems and its parallelizations.* PhD thesis, Michigan State University, 1991.