Algorithm 776: SRRIT: A Fortran Subroutine to Calculate the Dominant Invariant Subspace of a Nonsymmetric Matrix

Z. BAI
University of Kentucky
and
G. W. STEWART
University of Maryland

SRRIT is a Fortran program to calculate an approximate orthonormal basis for a dominant invariant subspace of a real matrix A by the method of simultaneous iteration. Specifically, given an integer m, SRRIT computes a matrix Q with m orthonormal columns and real quasi-triangular matrix T of order m such that the equation $AQ = QT$ is satisfied up to a tolerance specified by the user. The eigenvalues of T are approximations to the m eigenvalues of largest absolute magnitude of A, and the columns of Q span the invariant subspace corresponding to those eigenvalues. SRRIT references A only through a user-provided subroutine to form the product AQ; hence it is suitable for large sparse problems.

Categories and Subject Descriptors: D.3.2 [**Programming Languages**]: Language Classifications—*Fortran*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computations on matrices*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*eigenvalues*; G.4 [**Mathematics of Computing**]: Mathematical Software—*certification and testing*

General Terms: Algorithms

Additional Key Words and Phrases: Invariant subspace, nonsymmetric eigenvalue problem, project method

An earlier version appeared as Technical Report TR-154, Department of Computer Science, University of Maryland, 1978. Z. Bai was supported in part by NSF grant ASC-9313958 and in part by DOE grant DE-FG03-94ER25219, and G. W. Stewart was supported by the National Science Foundation under contract number CCR9115586.
Authors' addresses: Z. Bai, Department of Mathematics, University of Kentucky, Lexington, KY 40506; email: bai@ms.uky.edu; G. W. Stewart, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742; email: stewart@cs.umd.edu.

1. DESCRIPTION

The program described in this article is designed primarily to solve eigenvalue problems involving large, sparse nonsymmetric matrices. The program calculates a set of the eigenvalues of largest absolute magnitude of the matrix in question. In addition it calculates an orthonormal basis for the invariant subspace spanned by eigenvectors and principal vectors corresponding to the set of eigenvalues. No explicit representation of the matrix is required; instead the user furnishes a subroutine to calculate the product of the matrix with vectors.

Since the programs do not produce a set of eigenvectors corresponding to the eigenvalues computed, it is appropriate to begin with a mathematical description of what is actually computed and how the user may obtain eigenvectors from the output if they are required. Let A be a matrix of order n with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ ordered so that

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|.$$

An *invariant subspace* of A is any subspace $\mathcal{Q}$ for which

$$x \in \mathcal{Q} \Rightarrow Ax \in \mathcal{Q},$$

i.e., the subspace is transformed into itself by the matrix A.

If $\mathcal{Q}$ is an invariant subspace of A and the columns of $Q = (q_1, q_2, \ldots, q_m)$ form a basis for $\mathcal{Q}$, then $Aq_i \in \mathcal{Q}$, and hence Aq_i can be expressed as a linear combination of the columns of Q, i.e., there is an m-vector t_i such that $Aq_i = Qt_i$. Setting

$$T = (t_1, t_2, \ldots, t_m),$$

we have the relation

$$AQ = QT. \tag{1}$$

In fact the matrix T is just the representation of the matrix A in the subspace $\mathcal{Q}$ with respect to the basis Q. If x is an eigenvector of T corresponding to the eigenvalue λ, then it follows from (1) and the relation $Tx = \lambda x$ that

$$A(Qx) = \lambda(Qx), \tag{2}$$

so that Qx is an eigenvector of A corresponding to the eigenvalue λ. Thus the eigenvalues of T are also eigenvalues of A. Conversely, any eigenvalue of A whose eigenvector lies in $\mathcal{Q}$ is also an eigenvalue of T. Consequently, there is a one-one correspondence of eigenvectors of T and eigenvectors of A that lie in $\mathcal{Q}$.

If $|\lambda_i| > |\lambda_{i+1}|$, then there is a unique *dominant invariant subspace* $\mathcal{Q}_i$ corresponding to $\lambda_1, \lambda_2, \ldots, \lambda_i$. When $\mathcal{Q}_i$ and $\mathcal{Q}_{i+1}$ exist, $\mathcal{Q}_i \subset \mathcal{Q}_{i+1}$. SRRIT computes a nested sequence of orthonormal bases of $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_m$. Specifically, if all goes well, the subroutine produces a matrix Q with orthonormal columns having the property that if $|\lambda_i| > |\lambda_{i+1}|$, then $q_1, q_2, \ldots, q_i$ span $\mathcal{Q}_i$.

The case where λ_{i-1} and λ_i are a complex conjugate pair, and hence $|\lambda_{i-1}| = |\lambda_i|$, is treated as follows. The matrix Q is calculated so that the matrix T in (1) is quasi-triangular, i.e., T is block triangular with 1×1 and 2×2 blocks on its diagonal. The 1×1 blocks of T contain the real eigenvalues of A and the 2×2 blocks contain conjugate pairs of complex eigenvalues. This arrangement enables us to work entirely with real numbers, even when some of the eigenvalues of T are complex. The existence of such a decomposition is a consequence of Schur's theorem [Stewart 1973].

The eigenvalues of the matrix T computed by the program appear in descending order of magnitude along its diagonal. For fixed i, let $Q^{|i} = (q_1, q_2, \ldots, q_i)$ and let $T^{|\bar{i}}$ be the leading principal submatrix of T of order i. Then if the ith diagonal entry of T does not begin a 2×2 block, we have

$$AQ^{|i} = Q^{|i}T^{|\bar{i}}.$$

Thus the first i columns of Q span the invariant subspace corresponding to the first i eigenvalues of T. When $|\lambda_i| > |\lambda_{i+1}|$ this is the unique dominant invariant subspace $\mathcal{Q}_i$. When $|\lambda_i| = |\lambda_{i+1}|$ the columns of $Q^{|i}$ span a dominant invariant subspace; but it is not unique, since there is no telling which comes first, λ_i or λ_{i+1}.

Any manipulations of A within the subspace $\mathcal{Q}$ corresponding to Q can be accomplished by manipulating the matrix T. For example,

$$A^kQ = QT^k,$$

so that if $f(A)$ is any function defined by a power series, we have

$$f(A)Q = Qf(T).$$

If the spectrum of A that is not associated with Q is negligible, considerable work can be saved by working with the generally much smaller matrix T in the coordinate system defined by Q. If explicit eigenvectors are desired, they may be obtained by evaluating the eigenvectors x_i of T and forming Qx_i. The program STREVC in LAPACK [Anderson et al. 1995] will evaluate the eigenvectors of a quasi-triangular matrix.

2. USAGE

SRRIT is a package in ANSI Fortran 77 to calculate the basis for $\mathcal{Q}_m$ described in Section 1. All subroutines in SRRIT are provided in both single and double precision versions of real data. The single precision version of the main subroutine of SRRIT is called SRRIT and the double precision is DSRRIT. The calling sequence of DSRRIT matches SRRIT. Specifically, the calling sequence for SRRIT is

```
CALL SRRIT (N, NV, M, MAXIT, ISTART, Q, LDQ, AQ, LDA, T,
           LDT, WR, WI, RSD, ITRSD, IWORK, WORK, LWORK,
           INFO, EPS, ATQ)
```

The complete descriptions of the arguments are documented in the program. The user can specify the number NV of desired eigenvalues, the number M of vectors in the simultaneous iteration, and the tolerance value EPS for the convergence test. SRRIT requires a minimum of 2M integer array workspace and $M^2 + 5M$ real array workspace. On the return, if INFO is set to

0: successful exit or

$-k$: if the kth argument had an illegal value or

1: Reorthogonalization fails (see subroutine ORTH) or

2: Schur-Rayleigh-Ritz step fails (see subroutine SRRSTP) or

3: The representation matrix T with respect to the base Q is singular (see subroutine COND) or

4: SRRIT method fails to converge after MAXIT number of iterations.

The user is required to furnish a subroutine to calculate the product AQ. The calling sequence for this subroutine is

```
CALL ATQ(N, L, M, Q, LDQ, AQ, LDA)
```

with

N (input) INTEGER
The order of the matrix A.

L, M (input) INTEGER
The numbers of the first and the last column of Q to multiply by the matrix A.

Q (input) REAL array, dimension (LDQ, M)
Contains the matrix Q.

AQ (output) REAL array, dimension (LDQ, M)
On return, columns L through M of AQ contain the product of the matrix A with columns L through M of the matrix Q.

A call to ATQ causes the iteration counter to be increased by one, so that the parameter MAXIT is effectively a limit on the number of calls to ATQ.[1]

The convergence criterion is described in detail in Sections 3 and 4. Essentially the matrices Q and T calculated by the program will satisfy

$$(A + E)Q^{|\mathtt{NV}} = Q^{|\mathtt{NV}}T^{|\overline{\mathtt{NV}}}, \tag{3}$$

where NV (on return) is the number of columns that have converged and E is of order $\mathtt{EPS}/\|A\|$. From this it can be seen that the well-conditioned eigenvalues of A should have approximately –log EPS correct decimal digits.

The rate of convergence of the ith column of Q depends on the ratio $|\lambda_{\mathtt{M}+1}/\lambda_i|$. For this reason it may be desirable to take the number of columns M of Q to be greater than the number of columns NV that one desires to compute. For example, if the eigenvalues A are 1.0, 0.9, 0.5, . . . , it will pay to take M = 2 or 3, even if only the eigenvector corresponding to 1.0 is desired.

Since SRRIT is designed primarily to calculate the dominant eigenvalues of a large matrix, no provisions have been made to handle zero eigenvalues. In particular, zero eigenvalues can cause the program to stop in the auxiliary subroutine ORTH.

SRRIT requires a number of auxiliary subroutines which are described in Section 4. It also requires the BLAS and LAPACK subroutines. Section 6 contains a list of all auxiliary subroutines.

SRRIT can be used as a black box, in which case the first NV vectors it returns will satisfy (3). (But note that if the algorithm fails to converge completely, on return NV will be less than the number of vectors originally requested.) This packaging of the algorithm has involved a number of ad hoc decisions. Although the authors have attempted to make such decisions in a reasonable manner, it is too much to expect that the program will perform efficiently on all distributions of eigenvalues. Consequently the program has been written in such a way that it can be easily modified by someone who is familiar with its details. The purpose of the next two sections is to provide the interested user with these details.

3. METHOD

The Schur vectors Q of A are computed by a variant of simultaneous iteration, which is a generalization of the power method for finding the dominant eigenvector of a matrix. The method has an extensive literature [Bauer et al. 1957; Clint and Jennings 1970; Jennings and Stewart 1971; Rutishauser 1970; Stewart 1969]. Rutishauser has published a program for symmetric matrices, from which many of the features in SRRIT have been

[1]Our conventions differ from the "common" conventions for sparse matrix-vector products. The subroutine ATQ gives the user the chance to calculate AQ with only one pass over the data structure defining A, with a corresponding saving of work.

drawn [Rutishauser 1969]. The present variant of the simultaneous iteration method has been analyzed in Stewart [1976a]. An earlier version appeared in Stewart [1978]. The other existing software implementing the method are LOPSI (ACM Algorithm 570, fully available in the Collected Algorithms from ACM (CALGO)) [Stewart and Jennings 1981], which is based on the work of Clint and Jennings [1970] and Jennings and Stewart [1971], and EB12 in the Harwell Subroutine Library [Duff and Scott 1993], which is analyzed in Saad [1984]. LOPSI computes eigenvectors directly, which can lead to numerical difficulties when the eigensystem is ill conditioned. EB12, like SRRIT, is based on the Schur decomposition, and in addition has a Chebyshev acceleration option. SRRIT has the advantages that it was designed explicitly with customization in mind and it is nonproprietary.

The iteration for computing Q may be described briefly as follows. Start with an $n \times m$ matrix Q_0 having orthonormal columns. Given Q_μ, form $Q_{\mu+1}$ according to the formula

$$Q_{\mu+1} = (AQ_\mu)R_{\mu+1}^{-1},$$

where $R_{\mu+1}$ is either an identity matrix or an upper triangular matrix chosen to make the columns of $Q_{\mu+1}$ orthonormal (just how often such an orthogonalization should be performed will be discussed below). If $|\lambda_m| > |\lambda_{m+1}|$, then under mild restrictions on Q_0 the column space of Q_μ approaches $\mathcal{Q}_m$.[2]

The individual columns of Q_μ will in general approach the corresponding columns of the matrix Q defined in Section 1; however the error in the ith column is proportional to $\max\{|\lambda_i/\lambda_{i-1}|^\mu, |\lambda_{i+1}/\lambda_i|^\mu\}$, and convergence may be intolerably slow. The process may be accelerated by the occasional application of a "Schur-Rayleigh-Ritz step" (from which SRRIT derives its name), which will now be described. Start with Q_μ just after an orthogonalization step, so that $Q_\mu^T Q_\mu = I$. Form the matrix

$$B_\mu = Q_\mu^T A Q_\mu,$$

and reduce it to ordered quasi-triangular form T_μ by an orthogonal similarity transformation Z_μ:

$$Z_\mu^T B_\mu Z_\mu = T_\mu \tag{4}$$

Finally overwrite Q_μ with $Q_\mu Z_\mu$.

The matrices Q_μ formed in this way have the following property. If $|\lambda_{i-1}| > |\lambda_i| > |\lambda_{i+1}|$, then under mild restrictions on Q_0 the ith column

[2]For the details see Stewart [1976a]. In the presence of rounding error any power method can fail if the eigensystem is sufficiently ill conditioned. This problem has been analyzed by Wilkinson [1965, p. 573].

$q_i^{(\mu)}$ of Q_μ will converge approximately linearly to the ith column q_i of Q with ratio $|\lambda_{m+1}/\lambda_i|$. Thus not only is the convergence accelerated, but the first columns of Q_μ tend to converge faster than the later ones.

A number of practical questions remain to be answered.

(1) How should one determine when a column of Q_μ has converged?

(2) Can one take advantage of the early convergence of some of the columns of Q_μ to save computations?

(3) How often should one orthogonalize the columns of the Q_μ?

(4) How often should one perform the SRR step described above?

Here we merely outline the answers to these questions. The details are given in Section 4.

3.1 Convergence

If $|\lambda_{i-1}| = |\lambda_i|$ or $|\lambda_i| = |\lambda_{i+1}|$, the ith column of Q_μ is not uniquely determined; and when $|\lambda_i|$ is close to $|\lambda_{i+1}|$ or $|\lambda_{i-1}|$, the ith column cannot be computed accurately. Thus a convergence criterion based on the ith column $q_i^{(\mu)}$ of Q_μ becoming stationary is likely to fail when A has equimodular eigenvalues. Accordingly we have adopted a different criterion which amounts to requiring that the relation (1) almost be satisfied. Specifically, let $t_i^{(\mu)}$ denote the ith column of T_μ in (4). Then the ith column of the Q_μ produced by the SRR step is said to have converged if the two-norm of the residual vector

$$r_i^{(\mu)} = Aq_i^{(\mu)} - Q_\mu t_i^{(\mu)} \tag{5}$$

is less than $|\theta_i| * \mathrm{EPS}$, where θ_i is the ith eigenvalue of T_μ, and EPS is a prescribed tolerance. If this criterion is satisfied for each column of Q_μ, then the residual matrix

$$R_\mu = AQ_\mu - Q_\mu T_\mu$$

will be small. This in turn implies that there is a small matrix $E_\mu = -R_\mu Q_\mu^T$ such that

$$(A + E_\mu)Q_\mu = Q_\mu T_\mu,$$

so that Q_μ and T_μ solve the desired eigenproblem for the slightly perturbed matrix $A + E_\mu$, provided only that some small eigenvalue of $A + E_\mu$ has not by happenstance been included in T_μ. To avoid this possibility we group nearly equimodular eigenvalues together and require that the average of their absolute values settle down before testing their residuals. In addition a group of columns is tested only if the preceding columns have all converged.

3.2 Deflation

The theory of the iteration indicates that the initial columns of the Q_μ will converge before the later ones. When this happens considerable computation can be saved by freezing these columns. This saves multiplying the frozen columns by A, orthogonalizing them when $R_{\mu+1} \neq I$, and work in the SRR step.

3.3 Orthogonalization

The orthogonalization of the columns of AQ_μ is a moderately expensive procedure, which is to be put off as long as possible. The danger in postponing orthogonalization is that cancellation of significant figures can occur when AQ_μ is finally orthogonalized, as it must be just before an SRR step. In Stewart [1976a] it is shown that one can expect no more than

$$t = j \ \log_{10} \kappa(T) \tag{6}$$

decimal digits to cancel after j iterations without orthogonalization (here $\kappa(T) = \|T\|\|T^{-1}\|$ is the condition number of T with respect to inversion). The relation (6) can be used to determine the number of iterations between orthogonalizations.

3.4 SRR Steps

The SRR step does not actually accelerate the convergence of the Q_μ; rather it unscrambles approximations to the columns of $\mathcal{Q}_m$ that are already present in the column space of Q_μ and orders them properly. Therefore, the only time an SRR step needs to be performed is when it is expected that a column has converged. Since it is known from the theory of the iteration that the residual in (5) tends almost linearly to zero, the iteration at which they will satisfy the convergence criterion can be predicted from their values at two iterations. As with convergence, this prediction is done in groups corresponding to nearly equimodular eigenvalues.

4. DETAILS OF SRRIT

In designing SRRIT, we have tried to make it easily modifiable. This has been done in two ways. First, we have defined a number of important control parameters and given them values at the beginning of the program. The knowledgeable user may alter these values to improve the efficiency of the program in solving particular problems. Second, a number of important tasks have been isolated in independent subroutines. This should make it easy to modify the actual structure of SRRIT, should the user decide that such radical measures are necessary. In this section we describe SRRIT in some detail and specify the action of control parameters and the supporting subroutines.

Here follows a list of the control parameters with a brief description of their functions and their default initial values.

INIT The number of initial iterations to be performed at the outset (5).

STPFAC A constant used to determine the maximum number of iterations before the next SRR step (1.5).

ALPHA A parameter used in predicting when the next residual will converge (1.0).

BETA Another parameter used in predicting when the next residual will converge (1.1).

GRPTOL A tolerance for grouping equimodular eigenvalues (10^{-3}).

CNVTOL A convergence criterion for the average value of a cluster of equimodular eigenvalues (10^{-4}).

ORTTOL The number of decimal digits whose loss can be tolerated in orthogonalization steps (2).

We now give an informal description of SRRIT as it appears in the algorithm section. The variable L points to the first column of Q that has not converged. The variable IT is the iteration counter. The variable NXTSRR is the iteration at which the next SRR step is to take place, and the variable IDORT is the interval between orthogonalizations.

SRRIT:
1. initialize control parameters
2. initialize
   1. IT = 0;
   2. L = 1;
   3. initialize Q as described by ISTART
3. **SRR**: loop
   1. perform an **SRR** step
   2. compute residuals RSD
   3. check convergence, resetting L if necessary
   4. if L > NV or IT ≥ MAXIT then leave **SRR**
   5. calculate NXTSRR
   6. calculate IDORT and NXTORT
   7. Q = AQ; IT = IT + 1
   8. **ORTH**: loop until IT = NXTSRR
      1. **POWER**: loop until IT = NXTORT
         1. AQ = AQ
         2. Q = AQ
         3. IT = IT+1

      end **POWER**
      2. orthogonalize Q
      3. NXTORT = min(NXTSRR, IT+IDORT)
      end **ORTH**

   end **SRR**
4. NV = L - 1

end SRRIT

The details of this outline are as follows:

—*Line 2.3*: If ISTART ≤ 0, then Q is initialized using the random-number generation function SLARND, then orthonormalized by ORTH. If ISTART $= 1$, then Q is supplied by the user and is orthogonalized by calling subroutine ORTH. If ISTART > 1, the initial orthonomalized Q is supplied by the user.

—*Line 3*: This is the main loop of the program. Each time, an SRR step is performed and convergence is tested.

—*Line 3.1*: The SRR step is performed by the subroutine SRRSTP, which performs an SRR step on columns L through M of Q. After forming AQ and $T = Q^T(AQ)$, SRRSTP calls BLAS 2 LAPACK routine SGEHD2 to reduce T to upper Hessenberg form, then the subroutine SLAQR3 is called to reduce T to ordered quasi-triangular form. The triangularizing transformation U is postmultiplied into Q and AQ. The new Q and AQ, as well as T and its eigenvalues are returned.

—*Line 3.2*: The residuals RSD are computed by the subroutine RESID. RSD contains the two-norm of the residuals (5) for columns L through M of Q. For a complex pair of eigenvalues, the average of the norms of their two residuals is returned.

—*Line 3.3*: The algorithm for determining convergence is the following: starting with the Lth eigenvalue, the subroutine GROUP is called to determine a group of nearly equimodular eigenvalues, as defined by the parameter GRPTOL. The same is done for the old eigenvalues from the last SRR step. If the groups have the same number of eigenvalues and the average value of the eigenvalues has settled down (as specified by CNVTOL), then the residuals are averaged and tested against EPS. If the test is successful, L is increased by the number in the group. Otherwise control is passed to statement 3.4.

—*Line 3.4*: Here two conditions for stopping SRRIT are tested.

—*Line 3.5*: The iteration at which the next SRR-step is to take place (NXTSRR) is determined as follows. NXTSRR is tentatively set equal to STPFAC*IT. If the number of eigenvalues in the new and old groups corresponding to the next set of unconverged eigenvalues is the same, the average of the norms of the residuals of each group ARSD is calculated. If ARSD is greater than or equal to old ARSD (denoted as OARSD), then NXTSRR = STPFAC*IT. Otherwise

$$\text{NXTSRR} = \min(\text{IT} + \text{ALPHA} + \text{BETA*IDSRR}, \text{STPFAC*IT})$$

where

$$\texttt{IDSRR} = (\texttt{ITORSD} - \texttt{ITRSD})\frac{\log(\texttt{ARSD/EPS})}{\log(\texttt{ARSD/OARSD})}$$

where ITRSD and ITORSD are the iteration numbers where the new RSD and old RSD are computed. Finally NXTSRR is constrained to be less than or equal to MAXIT.

—*Line 3.6*: The interval IDORT between orthogonalizations is computed from (6):

$$\texttt{IDORT} = \max(1, \texttt{ORTTOL}/\log_{10} \kappa(T)),$$

where the condition number $\kappa(T)$ is calculated by the external function COND. The next orthogonalization occurs at

$$\texttt{NXTORT} = \min(\texttt{IT} + \texttt{IDORT}, \texttt{NXTSRR}).$$

—*Line 3.7*: Since the SRR step computes a product AQ, the iteration count must be increased and AQ placed back in Q.

—*Line 3.8*: Loop on orthogonalizations.

—*Line 3.8.1*: Loop overwriting Q with the product AQ.

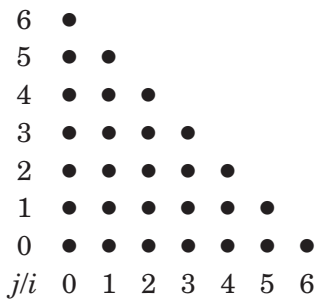
—*Line 3.8.2*: The subroutine ORTH is called to orthonormalize columns L through M of the array Q with respect to columns 1 through M. Columns 1 through L–1 are assumed to be orthonormalized. The method used is the modified Gram-Schmidt method with reorthogonalization. No more than MAXTRY reorthogonalizations are performed (currently, MAXTRY is set to 5), after which the routine executes a *stop*. The routine will also stop if any column becomes zero.

—*Line 4*: Set NV to the number of vectors that have actually converged and return.

5. NUMERICAL EXPERIMENTS

The program described above has been tested on a number of problems. In this section, we give three examples that illustrate the flexibility of the method and its ability to deal with equimodular or clustered eigenvalues. All the experiments have been run on a SUN Sparc workstation. We used a single-precision (mantissa of 32 bits) version of SRRIT for the presentation of numerical results.

Example. The first example is a random walk on an $(n + 1) \times (n + 1)$ triangular grid, which is illustrated below for $n = 6$.



The points of the grid are labeled (j, i), $(i = 0, \ldots, n, j = 0, \ldots, n - i)$. From the point (j, i), a transition may take place to one of the four adjacent points $(j + 1, i)$, $(j, i + 1)$, $(j - 1, i)$, $(j, i - 1)$. The probability of jumping to either of the nodes $(j - 1, i)$ or $(j, i - 1)$ is

$$pd(j, i) = \frac{j + i}{n} \tag{7}$$

with the probability being split equally between the two nodes when both nodes are on the grid. The probability of jumping to either of the nodes $(j + 1, i)$ or $(j, i + 1)$ is

$$pu(j, i) = 1 - pd(j, i). \tag{8}$$

with the probability again being split when both nodes are on the grid.

If the $(n + 1)(n + 2)/2$ nodes (j, i) are numbered $1,2, \ldots, (n + 1)(n + 2)/2$ in some fashion, then the random walk can be expressed as a finite Markov chain whose transition matrix A consisting of the probabilities a_{kl} of jumping from node l to node k (A is actually the transpose of the usual transition matrix; see Feller [1961]). To calculate the ith element of the vector Aq one need only regard the components of q as the average number of individuals at the nodes of the grid and use the probabilities (7) and (8) to calculate how many individuals will be at node i after the next transition.

We are interested in the steady state probabilities of the chain, which is ordinarily the appropriately scaled eigenvector corresponding to the eigenvalue unity. However, if we number the diagonals on the grid that are parallel to the hypotenuse by $0,1,2, \ldots, n$, then an individual on an even diagonal can only jump to an odd diagonal, and vice versa. This means that the chain is cyclic with period two, and that A has an eigenvalue of -1 as well as 1.

To run the problem on SRRIT, the nodes of the grid were matched with the components of the vector q in the order $(0,0), (1,0), \ldots, (n, 0), (0,1), (1,1), \ldots, (n - 1,1), (0,2), \ldots$. Note that the matrix A is

Table I. Numerical Results of the Random Walk Example

				IT = 0			
WR	=	9.547E-01	8.114E-02	-4.440E-02	3.793E-02	-1.019E-02	-1.019E-02
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	6.322E-03	-6.322E-03
RSD	=	2.534E-01	5.941E-01	5.856E-01	6.041E-01	6.015E-01	6.015E-01
NGRP	=	1					
CTR	=	9.547E-01	AE =	9.547E-01	ARSD =	2.534E-01	
NXTSRR	=	5		IDORT = 1			
				IT = 5			
WR	=	9.969E-01	3.230E-01	3.230E-01	-2.954E-01	-1.914E-01	2.835E-02
WI	=	0.000E+00	5.205E-02	-5.205E-02	0.000E+00	0.000E+00	0.000E+00
RSD	=	8.032E-02	8.593E-01	8.593E-01	8.675E-01	8.878E-01	9.221E-01
NGRP	=	1					
CTR	=	9.969E-01	AE =	9.969E-01	ARSD =	8.032E-02	
NXTSRR	=	7		IDORT = 1			
				IT = 7			
WR	=	9.985E-01	-4.240E-01	3.563E-01	-2.810E-01	2.145E-01	6.637E-02
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	6.545E-02	8.244E-01	9.003E-01	8.937E-01	8.756E-01	8.881E-01
NGRP	=	1					
CTR	=	9.985E-01	AE =	9.985E-01	ARSD =	6.545E-02	
NXTSRR	=	10		IDORT = 1			
				IT = 10			
WR	=	9.983E-01	4.873E-01	-4.041E-01	-3.772E-01	3.235E-01	1.557E-01
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	6.004E-02	8.040E-01	8.982E-01	8.660E-01	9.193E-01	9.263E-01
NGRP	=	1					
CTR	=	9.983E-01	AE =	9.983E-01	ARSD =	6.004E-02	
NXTSRR	=	15		IDORT = 2			
				IT = 15			
WR	=	9.973E-01	-6.102E-01	5.842E-01	4.607E-01	2.890E-01	-2.379E-01
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	4.633E-02	7.755E-01	7.253E-01	8.774E-01	8.990E-01	9.637E-01
NGRP	=	1					
CTR	=	9.973E-01	AE =	9.973E-01	ARSD =	4.633E-02	
NXTSRR	=	22		IDORT = 2			
				IT = 22			
WR	=	9.981E-01	-8.038E-01	7.285E-01	6.324E-01	3.781E-01	-3.540E-01
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	4.356E-02	5.885E-01	6.990E-01	7.443E-01	9.302E-01	8.976E-01
NGRP	=	1					
CTR	=	9.981E-01	AE =	9.981E-01	ARSD =	4.356E-02	
NXTSRR	=	33		IDORT = 2			
				IT = 33			
WR	=	1.000E+00	-9.398E-01	9.217E-01	8.784E-01	-5.927E-01	1.048E-01
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	2.765E-02	3.097E-01	3.701E-01	4.638E-01	7.542E-01	9.283E-01
NGRP	=	1					
CTR	=	1.000E+00	AE =	1.000E+00	ARSD =	2.765E-02	
NXTSRR	=	49		IDORT = 1			

never explicitly used; all computations are done in terms of the transition probabilities (7) and (8).

Table I. *Continued*

				IT = 49			
WR	=	1.000E+00	-9.870E-01	9.717E-01	9.000E-01	-7.130E-01	-4.267E-01
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	6.739E-03	1.101E-01	1.273E-01	4.398E-01	5.964E-01	8.724E-01
NGRP	=	1					
CTR	=	1.000E+00	AE 1 =	1.000E+00	ARSD =	6.739E-03	
NXTSRR	=	73		IDORT = 2			
				IT = 73			
WR	=	1.000E+00	-9.917E-01	9.867E-01	-9.631E-01	9.392E-01	-8.748E-01
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	2.695E-03	2.716E-02	3.938E-02	2.733E-01	2.978E-01	4.538E-01
NGRP	=	1					
CTR	=	1.000E+00	AE =	1.000E+00	ARSD =	2.695E-03	
NXTSRR	=	109		IDORT = 2			
				IT = 109			
WR	=	1.000E+00	-9.974E-01	-9.974E-01	9.924E-01	9.712E-01	-9.691E-01
WI	=	0.000E+00	8.061E-04	-8.061E-04	0.000E+00	0.000E+00	0.000E+00
RSD	=	1.207E-03	3.388E-02	3.388E-02	1.288E-02	1.060E-01	1.298E-01
NGRP	=	1					
CTR	=	1.000E+00	AE =	1.000E+00	ARSD =	1.207E-03	
NXTSRR	=	163		IDORT = 2			
				IT = 163			
WR	=	1.000E+00	-1.000E+00	-9.938E-01	9.934E-01	-9.753E-01	9.752E-01
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	9.870E-05	3.319E-03	2.206E-03	1.268E-03	2.643E-02	2.649E-02
NGRP	=	1					
CTR	=	1.000E+00	AE =	8.017E-06	ARSD =	2.348E-03	
NXTSRR	=	244		IDORT = 2			
				IT = 244			
WR	=	1.000E+00	-1.000E+00	-9.935E-01	9.935E-01	-9.755E-01	9.755E-01
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	1.738E-06	5.403E-05	6.572E-05	3.626E-05	3.097E-03	3.296E-03
NGRP	=	2					
CTR	=	1.000E+00	AE =	5.960E-08	ARSD =	3.823E-05	
NXTSRR	=	274		IDORT = 2			
				IT = 274			
WR	=	1.000E+00	-1.000E+00	9.935E-01	-9.935E-01	-9.755E-01	9.755E-01
WI	=	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00
RSD	=	2.440E-06	1.188E-05	9.631E-06	1.754E-05	1.422E-03	1.522E-03
NGRP	=	2					
CTR	=	1.000E+00	AE =	1.371E-06	ARSD =	8.576E-06	
NGRP	=	2					
CTR	=	9.935E-01	AE =	3.576E-07	ARSD =	1.415E-05	

The problem was run for a 30×30 grid which means $N = 496$. We took $\mathtt{M} = 6$, $\mathtt{NV} = 4$, and $\mathtt{EPS} = 10^{-5}$. The results for each iteration in which an SRR step was performed are summarized in Table I. The variables WR and WI are the real and imaginary parts of the eigenvalues. RSD is the norm of the corresponding residual. CTR is the center of the current convergence cluster. AE is the average value of the eigenvalues in the cluster. ARSD is

Table II. Number of Iterations

m	it	$m \times it$
2	1660	3320
4	600	2400
6	320	1920
8	183	1464

the average of the residuals ARSD. NXTSRR is the number of iterations to the next SRR step and IDORT is the number to the next orthogonalization.

The course of the iteration is unexceptionable. The program doubles the interval between SRR steps until it can predict convergence of the first cluster corresponding to the eigenvalues ± 1. The first prediction falls slightly short, but the second gets it. The program terminates on the convergence of the second group of two eigenvalues.

To compare the actual costs, runs were made with $m = 2,4,6,8$. Table II shows the number of iterations for the convergence of the first group of two eigenvalues.

As predicted by the convergence theory, the number of iterations decreases as m increases. However, as m increases we must also multiply more columns of Q by A, and for this particular problem the number of matrix-vector multiplications $m \times it$ is probably a better measure of the amount of work involved. From the table it is seen that this measure is also decreasing, although less dramatically than the number of iterations. This of course does not include the overhead generated by SRRIT itself, which increases with m and may be considerable.

Example. This example shows how SRRIT can be used in conjunction with the inverse power method to find the smallest eigenvalues of a matrix. Consider the boundary value problem

$$y'' + \mu^2 y = 0,$$

$$y(0) = 0, y'(0) + \gamma y'(1) = 0, \ 0 < \gamma < 1. \tag{9}$$

The eigenvalues of this problem are easily seen to be given by $\mu = icosh^{-1}(-\gamma^{-1})$, which are complex. Table III lists the reciprocals of the first eight eigenvalues for $\gamma = 0.01$.

The solution of (9) can be approximated by finite-difference techniques as follows. Let y_j denote the approximate solution at the point $x_j = j/(n + 1)$ $(j = 0,1, \ldots, n + 1)$. Replacing the derivatives in (9) with three point difference operators, we obtain the following $(n + 1)$-by-$(n + 1)$ generalized matrix eigenvalue problem for $y = (y_1, y_2, \ldots, y_{n+1})^T$:

$$Ay + \mu^2 By = 0,$$

Table III. The Reciprocals of Five Eigenvalues of Largest Absolute Magnitude

μ^{-2}	$\lvert\mu^{-2}\rvert$
-0.012644 $\pm$0.02313i	0.02636
0.004446 $\pm$0.00739i	0.00854
0.002895 $\pm$0.00220i	0.00364
0.001274 $\pm$0.00089i	0.00195

where

$$A = \begin{pmatrix} -2 & 1 & & & & & & \\ 1 & -2 & 1 & & & & & \\ & 1 & -2 & 1 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & 1 & -2 & 1 & \\ & & & & & 1 & -2 & 1 \\ 4 & -1 & & & & \gamma & -4\gamma & 3\gamma \end{pmatrix}$$

and $B = h^2\text{diag}(1,1, \ldots, 1,0)$. We may recast this problem in the form

$$Cy = -\frac{1}{\mu^2}y,$$

where $C = A^{-1}B$.

To apply SRRIT to this problem, we must be able to compute $z = Cq$ for any vector q. This can be done by solving the linear system

$$Az = Bq,$$

which is done by sparse Gaussian elimination.

The problem was run for $n = 301$ with M = 6, NV = 4, and EPS = 10^{-5}. The results are shown in Table IV.

Given the extremely favorable ratios of the eigenvalues in Table IV—the absolute value of the ratio of the seventh to the first is about 0.075—it is not surprising that the iteration converges quickly. Indeed the only thing preventing convergence at the fifth iteration is that the first eigenvalue changed from real in the first iteration to complex in the fifth. Thus the problem is hardly a fair test of the SRRIT machinery. However, it is an excellent example of how easy it is to apply SRRIT to a problem with complex eigenvalues. It also disposes of the notion that large eigenvalue problems must always require a large amount of work to solve: the factor that limits the size is the storage available, not the time required to compute Ax. The next example from partial differential equations demonstrates this point again.

Table IV. Numerical Results of the Boundary Value Problem

				IT = 0			
WR	=	-1.528E-01	3.013E-04	3.013E-04	1.455E-04	1.004E-04	2.828E-05
WI	=	0.000E+00	1.064E-04	-1.064E-04	0.000E+00	0.000E+00	0.000E+00
RSD	=	1.434E-01	1.094E-02	1.094E-02	2.704E-03	6.453E-03	6.423E-03
NGRP	=	1					
CTR	=	1.528E-01	AE =	-1.528E-01	ARSD =	1.434E-01	
NXTSRR	=	5		IDORT = 1			
				IT = 5			
WR	=	-1.264E-02	-1.264E-02	4.429E-03	4.429E-03	2.542E-03	2.542E-03
WI	=	2.312E-02	-2.312E-02	7.288E-03	-7.288E-03	2.575E-03	-2.575E-03
RSD	=	1.115E-07	1.115E-07	2.912E-05	2.912E-05	6.126E-04	6.126E-04
NGRP	=	2					
CTR	=	2.636E-02	AE =	-1.264E-02	ARSD =	1.115E-07	
NXTSRR	=	7		IDORT = 1			
				IT = 7			
WR	=	-1.264E-02	-1.264E-02	4.446E-03	4.446E-03	2.904E-03	2.904E-03
WI	=	2.312E-02	-2.312E-02	7.309E-03	-7.309E-03	2.316E-03	-2.316E-03
RSD	=	2.000E-08	2.000E-08	1.994E-06	1.994E-06	2.358E-04	2.358E-04
NGRP	=	2					
CTR	=	2.636E-02	AE =	-1.264E-02	ARSD =	2.000E-08	
NGRP	=	2					
CTR	=	8.555E-03	AE =	4.446E-03	ARSD =	1.994E-06	
NXTSRR	=	8		IDORT = 1			
				IT = 8			
WR	=	-1.264E-02	-1.264E-02	4.447E-03	4.447E-03	2.888E-03	2.888E-03
WI	=	2.312E-02	-2.312E-02	7.308E-03	-7.308E-03	2.254E-03	-2.254E-03
RSD	=	2.000E-08	2.000E-08	4.426E-07	4.426E-07	1.292E-04	1.292E-04
NGRP	=	2					
CTR	=	8.555E-03	AE =	4.447E-03	ARSD =	4.426E-07	
NXTSRR	=	9		IDORT = 1			
				IT = 9			
WR	=	-1.264E-02	-1.264E-02	4.447E-03	4.447E-03	2.886E-03	2.886E-03
WI	=	2.312E-02	-2.312E-02	7.309E-03	-7.309E-03	2.242E-03	-2.242E-03
RSD	=	2.000E-08	2.000E-08	8.069E-08	8.069E-08	6.098E-05	6.098E-05
NGRP	=	2					
CTR	=	8.555E-03	AE =	4.447E-03	ARSD =	8.069E-08	
NGRP	=	2					
CTR	=	3.654E-03	AE =	2.886E-03	ARSD =	6.098E-05	

Example. Let us consider the following model convection-diffusion problem:

$$-\Delta u + 2p_1 u_x + 2p_2 u_y - p_3 u = f(x, y) \text{ in } \Omega$$

$$u = g(x, y) \text{ on } \partial\Omega$$

where Ω is the unit square $\{(x, y) \in \mathbf{R}^2, 0 \le x,y \le 1\}$, and p_1, p_2, and p_3 are positive constants. f and g are given functions. After discretizing the equation by the standard five-point centered differences on a uniform $n \times n$ grid, we get a nonsymmetric $n^2 \times n^2$ block tridiagonal matrix

Table V. Number of Iterations

m	λ_{m+1}/λ_1	it	$m \times it$
2	0.9964	1280	2560
4	0.9904	593	2372
6	0.9868	320	1920

$$A = \begin{pmatrix} B & (\beta-1)I & & & \\ (-\beta+1) & B & (\beta-1)I & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & (\beta-1)I \\ & & & (-\beta+1)I & B \end{pmatrix}$$

with

$$B = \begin{pmatrix} 4-\sigma & \gamma-1 & & & \\ -\gamma-1 & 4-\sigma & \gamma-1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \gamma-1 \\ & & & -\gamma-1 & 4-\sigma \end{pmatrix},$$

where $\beta = p_1 h$, $\gamma = p_2 h$, $\sigma = p_3 h^2$, and $h = 1/(n+1)$. The eigenvalues of matrix A are given by

$$\lambda_{kl} = 4 - \sigma + 2(1-\beta^2)^{1/2}\cos\frac{k\pi}{n+1} + 2(1-\gamma^2)^{1/2}\cos\frac{l\pi}{n+1}, \quad 1 \le k,l \le n.$$

The following lists the first 10 dominant eigenvalues (in seven decimal digits) for $p_1 = p_2 = p_3 = 1$:

7.977818, 7.949033, 7.949033, 7.920248, 7.901366,
7.901366, 7.872581, 7.872581, 7.835278, 7.835278.

The algorithm was run on the 961×961 matrix A obtained by taking a 31×31 mesh grid. We are interested in the first dominant eigenvalue. The results obtained are listed in Table V for different values of m (EPS $= 10^{-4}$). The eigenvalues in this problem are tightly clustered, and increasing the value of m significantly decreases the number of iterations.

6. LIST OF SUBROUTINES CALLED BY SRRIT

Program SRRIT comes with the following subroutines:

SRRSTP Performs an Schur-Rayleigh-Ritz iteration step.

ORTH Orthonormalizes columns of a matrix.

RESID Computes the column norms of residual vectors $R = AQ - QT$.

GROUP Finds a cluster of complex numbers.

SLAQR3 Computes the Schur factorization of a real upper Hessenberg matrix. The blocks of quasi-triangular forms are ordered so that the eigenvalues appear in descending order of absolute value along the diagonal. The decomposition produced by SLAQR3 differs from the one produced by EISPACK subroutine HQR [Smith et al. 1974] or LAPACK subroutine SHSEQR in that the eigenvalues of the final quasi-triangular matrix are ordered. It is essentially the same as the program HQR3 [Stewart 1976b]. However, instead of using QR iteration to do the diagonal swapping in HQR3, SLAQR3 uses a direct swapping method [Bai and Demmel 1993].

COND Estimates the l_∞-norm condition number with respect to inversion of an upper Hessenberg matrix.

SLARAN Generates a random real number from a uniform (0,1) distribution.

In addition, the following subroutines from standard BLAS and LAPACK are also used in SRRIT:

—Subroutines from BLAS

Level 1: ISAMAX, SCOPY, SDOT, SROT, SAXPY, SSCAL, SSWAP, SNRM2
Level 2: SGEMV, SGER, SSYR, STRMV, STBSV, LSAME
Level 3: SGEMM, STRMM

—Subroutines from LAPACK

SGEHRD, SGEHD2, SLAEXC, SLARFG, SLARF, SLARFX, SLASY2, SLANV2, SLAPY2, SLARTG, SLANGE, SLANHS, SLASSQ, SLACPY, SLAMCH, XERBLA, SLABAD, SLASET, SORGHR, SORGQR, SLARFB, SLAHRD, SLARFT, ILAENV, SORG2R.

ACKNOWLEDGMENTS

The authors would like to express thanks to the referees and the editor John Reid, whose comments led to improvements in the presentation and program.

REFERENCES

ANDERSON, E., BAI, Z., BISCHOF, C. H., DEMMEL, J., DONGARRA, J. J., DU CROZ, J., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., OSTROUCHOV, S., AND SORENSEN, D. C. 1995. *LAPACK User's Guide*. 2nd ed. Society for Industrial and Applied Mathematics, Philadelphia, PA.

BAI, Z. AND DEMMEL, J. 1993. On swapping diagonal blocks in real Schur form. *Lin. Alg. Appl. 186*, 73–95.

BAUER, F. L. 1957. Das verfahren der treppeniteration und verwandte verfahren zur lösung algebraischer eigenwertprobleme. *Z. Angew. Math. Phys. 8*, 214–235.

CLINT, M. AND JENNINGS, A. 1970. The evaluation of eigenvalues and eigenvectors of a real symmetric matrix by simultaneous iteration. *Comput. J. 13*, 68–80.

DUFF, I. S. AND SCOTT, J. A. 1993. Computing selected eigenvalues of sparse unsymmetric matrices using subspace iteration. *ACM Trans. Math. Softw. 19*, 2 (June), 137–159.

FELLER, W. 1961. *An Introduction to Probability Theory and Its Applications*. John Wiley & Sons, Inc., New York, NY.

JENNINGS, A. AND STEWART, W. J. 1971. A simultaneous iteration method for the unsymmetric eigenvalue problem. *J. Inst. Math. Appl. 8*, 111–121.

RUTISHAUSER, H. 1969. Computational aspects of F. L. Bauer's simultaneous iteration method. *Numer. Math. 13*, 4–13.

RUTISHAUSER, H. 1971. Simultaneous iteration method for symmetric matrices. *Numer. Math. 16*, 205–223.

SAAD, Y. 1984. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Math. Comput. 42*, 71–90.

SMITH, B. T., BOYLE, J. M., GARBOW, B. S., IKEBE, Y., KLEMA, V. C., AND MOLER, C. B. 1974. *Matrix Eigensystem Routines: EISPACK Guide*. Springer Lecture Notes in Computer Science, vol. 6. Springer-Verlag, Berlin, Germany.

STEWART, G. W. 1969. Accelerating the orthogonal iteration for the eigenvalues of a Hermitian matrix. *Numer. Math. 13*, 362–376.

STEWART, G. W. 1973. *Introduction to Matrix Computations*. Academic Press, Inc., Orlando, FL.

STEWART, G. W. 1976a. Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices. *Numer. Math. 25*, 123–126.

STEWART, G. W. 1976b. ALGORITHM 506: HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix. *ACM Trans. Math. Softw. 2*, 3 (Sept.), 275–280.

STEWART, G. W. 1978. SRRIT: A FORTRAN subroutine to calculate the dominant invariant subspace of a nonsymmetric marix. Tech. Rep. TR-154. Department of Computer Science, University of Maryland, College Park, MD.

STEWART, W. J. AND JENNINGS, A. 1981. ALGORITHM 570: LOPSI: A simultaneous iteration algorithm for real matrices. *ACM Trans. Math. Softw. 7*, 2 (June), 230–232.

WILKINSON, J. H. 1965. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, UK.

Received: May 1994; revised: October 1994; accepted: March 1997