University of Maryland College Park

Institute for Advanced Computer Studies TR–92–61
Department of Computer Science TR–2908

SRRIT — A FORTRAN Subroutine to Calculate the Dominant Invariant Subspace of a Nonsymmetric Matrix*

Z. Bai†
G. W. Stewart‡

May, 1992

ABSTRACT

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 [12]. Specifically, given an integer m, *SRRIT* attempts to compute 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 largest eigenvalues 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.

*This report is available by anonymous ftp from `thales.cs.umd.edu` in the directory `pub/reports`. The program is available in `pub/srrit`

†Department of Mathematics, University of Kentucky, Lexington, KY 40506.

‡Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. This work was supported in part by the National Science Foundation under Contract Number CCR9115586.

SRRIT — A FORTRAN Subroutine to Calculate the Dominant Invariant Subspace of a Nonsymmetric Matrix*

Z. Bai†
G. W. Stewart‡

Abstract

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 [12]. Specifically, given an integer m, *SRRIT* attempts to compute 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 largest eigenvalues 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.

1. Description

The program described in this paper is designed primarily to solve eigenvalue problems involving large, sparse nonsymmetric matrices. The program attempts to calculate a set of the largest eigenvalues of the matrix in question. In addition it calculates a canonical 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 a vector.

---

*The report is available by anonymous ftp from `thales.cs.umd.edu` in the directory `pub/reports`. The program is available in `pub/srrit`. Earlier version appeared as Technical Report TR-154, Department of Computer Science, University of Maryland, 1978.

†Department of Mathematics, University of Kentucky, Lexington, KY 40506.

‡Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland 20742. This work was supported in part by the National Science Foundation under Contract Number CCR9115586.

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 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

$$x \in \mathcal{Q} \implies Ax \in \mathcal{Q};$$

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 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 eigenvector 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* attempts to compute 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 structure of a typical quasi-triangular matrix is illustrated below for $m = 6$:

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{pmatrix}.$$

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 [11].

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 $T^{\overline{|i}}$ be the leading principal submatrix of T of order i. Then if the ith diagonal entry of T does not begin a 2×2 blocks, we have

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

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,

$$A^k Q = QT^k,$$

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 of T and appling (2). The program `STREVC` in LAPACK [1] will evaluate the eigenvectors of a quasi-triangular matrix.

2. Usage

SRRIT is a subroutine in ANSI FORTRAN 77 to calculate the basis for $\mathcal{Q}_m$ described in Section 1. 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 )
```

with

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

`NV` (input) INTEGER
`NV` is the size of the leading invariant subspace of A that the user desired.

`M` (input) INTEGER
`M` is the size of iteration space ($\mathtt{NV} \leq \mathtt{M} \leq \mathtt{N}$).

`MAXIT` (input) INTEGER
`MAXIT` is an upper bound on the number of iterations the program is to execute.

`ISTART` (input) INTEGER
`ISTART` specifies whether user supplies an initial basis `Q`.
≤ 0, `Q` is initialized by the program.
$= 1$, starting `Q` has been set in the input but is not orthonormal.
> 1, starting `Q` has been set in the input and is orthonormal.

`Q` (input/output) REAL array, dimension(`LDQ, M`)
On entry, if $\mathtt{ISTART} > 0$, `Q` contains the starting Q which will be used in the simultaneous iteration. On exit, `Q` contains the orthonormal vectors described above.

`LDQ` (input) INTEGER
The leading dimension of `Q`, $\mathtt{LDQ} \geq \max(\mathtt{1}, \mathtt{N})$.

`AQ` (output) REAL array, dimension(`LDA, M`)
On exit, `AQ` contains the product AQ.

LDA (input) INTEGER
The leading dimension of A, LDA $\geq$ max(1,N).

T (output) REAL array, dimension(LDT, M)
On exit, T contains of representation of A described above.

LDT (input) INTEGER
The leading dimension of T, LDT $\geq$ max(1,M).

WR,WI (output) REAL arrays, dimension (M)
On exit, WR and WI contain the real and imaginary parts, respectively, of the eigenvalues of T, which is also the dominant eigenvalues of matrix A. The eigenvalues appear in decreasing order.

RSD (output) REAL arrays, dimension(M)
On exit, RSD contains the 2-norm of the residual vectors.

ITRSD (output) INTEGER array, dimension(M)
On exit, ITRSD contains the iteration numbers at which the residuals were computed.

IWORK (workspace) INTEGER array, dimension(2*M)

WORK (workspace) REAL array, dimension(LWORK)

LWORK (input) INTEGER
The length of work space. LWORK $>=$ M $*$ M $+$ 5 $*$ M.

INFO (output) INTEGER
On exit, if INFO is set to
- 0: normal return.
- 1: error from initial orthogonalization
- 2: error from subroutine SRRSTP
- 3: error from subroutine COND
- 4: error from orthogonalization in power iteration

`EPS` (input) REAL
A convergence criterion supplied by user.

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` contains 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`.*

The convergence criterion is described in detail in section 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 that the well-conditioned eigenvalues of A should have approximately $-\log \mathtt{EPS}$ correct decimal digits.

The rate of convergence of the ith column of Q depends on the ratio $|\lambda_{\mathtt{M+1}}/\lambda_i|$. From this reason it may be desirable to take the number of columns `M` of Q to be

*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.

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 largest 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 subroutine (SRRSTP, RESID, GROUP, ORTH, COND) which are described in Section 5. It also requires the LAPACK subroutines such as SGEHD2, and the some variation of the LAPACK subroutines such as SLAQR3 etc. Appendix A contains list of all auxiliary subroutines.

SRRIT can be used as a black box. As such the first NV vectors it returns will satisfy (3), although not as many as vectors as the user requests need have converged by the time MAXIT is reached. However, the construction of the program 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 three 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 [3, 4, 5, 8, 10], and Rutishauser [7] has published a program for symmetric matrices, from which many of the features in *SRRIT* have been drawn. The present variant of simultaneous iteration method has been analyzed in [12].

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 $\mathcal{Q}_m$.

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 $Q_\mu^{\mathrm{T}} Q_\mu = I$. Form the matrix

$$B_\mu = Q_\mu^{\mathrm{T}} A Q_\mu,$$

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

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

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

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 $q_i^{(\mu)}$ 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 shall merely outline the answers to these questions. The details will be given in the next section.

1. *Convergence.* If $|\lambda_{i-1}| = |\lambda_i|$ or $|\lambda_i| = |\lambda_{i+1}|$, the ith column of Q 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 2-norm of the residual vector

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

is less than some 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 $E_\mu = -R_\mu Q_\mu^{\mathrm{T}}$ 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.

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. *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 [12] 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 condition number of T with respect to inversion). The relation (6) can be used to determine the number of iterations between orthogonalizations.

4. *SRR Steps.* The SRR step described above does not actually accelerate the convergence of the Q_μ; rather it unscramble 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 shall describe *SRRIT* in some detail, specifying the action of control parameters. In the next section we shall describe the supporting subroutines.

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

`INIT` A number of initial iteration to be performed at the outset (5).

`STPFAC` A constant used to determine the maximum number of iterations before the next SRR step (2).

`ALPAH` 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^{-3}).

`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 orthogonalization.

SRRIT:

1. initialize control parameters
2. initialize
   1. `IT` = 0;
   2. `L` = 0;
   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` $\geq$ `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 (the numbers correspond to the statements in the algorithm).

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 user, and is orthogonalized by calling subroutine `ORTH`. If `ISTART` > 1, the initial orthonomalized `Q` is supplied by user.

3. This is the main loop of the program. Each time an SRR step is performed and convergence is tested.

3.1. The SRR step is performed by the subroutine `SRRSTP`, which returns the new `Q` and `AQ`, as well as `T` and its eigenvalues.

3.2. The residuals `RSD` are computed by the subroutine `RESID`.

3.3. The algorithm for determining convergence is the following, starting with the `L`-th 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 successful. `L` is increased by the number in the group, and the tests are repeated. Otherwise control is passed to statement 3.4.

3.4. Here two conditions for stopping *SRRIT* are tested.

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 or equal to old `ARSD` (denoted as `OARSD`), then `NXTSRR` = `STPFAC*IT`. Otherwise

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

where

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

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`.

3.6. The interval `IDORT` between orthogonalizations is computed from (6):

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

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

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

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

3.8. Loop on orthogonalizations.

3.8.1. Loop overwriting `Q` with the product `AQ`.

4. Set `NV` to the number of vectors that have actually converged and return.

5. Auxiliary Subroutines

In this section we shall describe some of the subroutines called by *SRRIT*. All the required subroutines and their corresponding functionalities are listed in Appendix A. These subroutines have been coded in greater generality than is strictly required by *SRRIT* in order to make the program easily modifiable by the user.

```
SRRSTP( N, L, M, Q, LDQ, AQ, LDA, T, LDT, WR, WI, U, LDU, WORK,
        LWORK, INFO )
```

This subroutine performs an SRR step on columns `L` through `M` of `Q`. After forming AQ and $T = Q^{\mathrm{T}}(AQ)$, the routine 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 computed eigenvalues are placed in the arrays `WR`, `WI`.

```
RESID( N, L, M, Q, LDQ, AQ, LDA, T, LDT, RSD )
```

This subroutine computes the 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.

```
GROUP( L, M, WR, WI, RSD, NGRP, CTR, AE, ARSD, GRPTOL )
```

This subroutine locates a group of approximately equimodular eigenvalues $\lambda_{\mathtt{L}}$, $\lambda_{\mathtt{L+1}}$, $\ldots, \lambda_{\mathtt{N+NGRP-1}}$. The eigenvalues so grouped satisfy

$$||\lambda_i| - \mathtt{CTR}| \leq \mathtt{GRPTOL} * \mathtt{CTR}, \qquad i = \mathtt{L}, \mathtt{L}+1, \ldots, \mathtt{L}+\mathtt{NGRP}-1.$$

The mean of the group is returned in `AE`.

```
ORTH( N, L, M, Q, LDQ, INFO )
```

This subroutine orthonormalizes column `L` through `M` of the array `Q` with respect to column `1` through `M`. Column `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.

```
SLAQR3( IJOB, ICOMPZ, N, ILO, IHI, H, LDH, WR, WI, Z, LDZ, WORK,
                              INFO )
```

This subroutine 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` [9] 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` [13]. However, instead of using QR iteration to do the diagonal swapping in `HQR3`, `SLAQR3` uses a direct swapping method [2].

6. 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 1+ workstation. We used single precision (mantissa of 32 bits).

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

$$\begin{array}{cccccccc}
6 & \bullet & & & & & & \\
5 & \bullet & \bullet & & & & & \\
4 & \bullet & \bullet & \bullet & & & & \\
3 & \bullet & \bullet & \bullet & \bullet & & & \\
2 & \bullet & \bullet & \bullet & \bullet & \bullet & & \\
1 & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \\
0 & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
j/i & 0 & 1 & 2 & 3 & 4 & 5 & 6
\end{array}$$

The points of the grid are labelled $(j, i), (i = 0, \ldots, n, j = 0, \ldots, n - i)$. 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 [6]). 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 never explicitly used; all computations are done in terms of the transition probabilities (7) and (8).

The problem was run for a 30×30 grid which means $N = 496$. We took `M = 6`, `NV = 4`, and `EPS =` 10^{-5} . The results for each iteration for each iteration in which an SRR step was performed are summarized in the following. 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 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.

```
                                 IT =    0
WR    =  0.8225E-01 -0.5044E-01 -0.1708E-02 -0.1708E-02  0.2715E-01 -0.2220E-01
WI    =  0.0000E+00  0.0000E+00  0.1173E-01 -0.1173E-01  0.0000E+00  0.0000E+00
RSD   =  0.5798E+00  0.6257E+00  0.8696E+00  0.8696E+00  0.5774E+00  0.5797E+00
NGRP =     1
CTR   =  0.8225E-01
```

```
AE   =  0.8225E-01
ARSD =  0.5798E+00
NXTSRR =   5  IDORT  =   1
                              IT =    5
WR   = -0.4445E+00 -0.3217E+00  0.2972E+00  0.1818E+00 -0.1370E+00 -0.2263E-01
WI   =  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
RSD  =  0.7679E+00  0.8694E+00  0.8836E+00  0.8691E+00  0.9538E+00  0.8957E+00
NGRP =    1
CTR  =  0.4445E+00
AE   = -0.4445E+00
ARSD =  0.7679E+00
NXTSRR =  10  IDORT  =   1
                              IT =   10
WR   = -0.7853E+00 -0.6389E+00  0.4249E+00 -0.3609E+00  0.1900E+00 -0.7887E-01
WI   =  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
RSD  =  0.6394E+00  0.7446E+00  0.7923E+00  0.9019E+00  0.9719E+00  0.9758E+00
NGRP =    1
CTR  =  0.7853E+00
AE   = -0.7853E+00
ARSD =  0.6394E+00
NXTSRR =  20  IDORT  =   1
                              IT =   20
WR   = -0.9179E+00  0.6101E+00 -0.5658E+00  0.3678E+00 -0.3665E+00 -0.1833E+00
WI   =  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
RSD  =  0.3907E+00  0.7700E+00  0.7185E+00  0.9397E+00  0.8234E+00  0.9254E+00
NGRP =    1
CTR  =  0.9179E+00
AE   = -0.9179E+00
ARSD =  0.3907E+00
NXTSRR =  40  IDORT  =   2
                              IT =   40
WR   = -0.9891E+00  0.9585E+00 -0.8963E+00  0.8758E+00 -0.5805E+00  0.1108E+00
WI   =  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
RSD  =  0.2900E-01  0.2592E+00  0.4044E+00  0.4707E+00  0.7484E+00  0.9456E+00
NGRP =    1
CTR  =  0.9891E+00
AE   = -0.9891E+00
ARSD =  0.2900E-01
NXTSRR =  80  IDORT  =   1
                              IT =   80
WR   = -0.9990E+00  0.9968E+00  0.9913E+00 -0.9907E+00 -0.9579E+00  0.8811E+00
WI   =  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
RSD  =  0.2347E-01  0.4834E-01  0.3555E-01  0.2970E-01  0.1595E+00  0.4273E+00
NGRP =    1
CTR  =  0.9990E+00
```

```
AE   = -0.9990E+00
ARSD =  0.2347E-01
NXTSRR = 160  IDORT  =  15
                              IT =  160
WR   = -0.1000E+01  0.1000E+01  0.9934E+00 -0.9934E+00 -0.9754E+00  0.9746E+00
WI   =  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
RSD  =  0.5815E-03  0.3167E-02  0.2486E-02  0.6028E-03  0.1128E-01  0.4427E-01
NGRP =    2
CTR  =  0.1000E+01
AE   = -0.1884E-04
ARSD =  0.2277E-02
NXTSRR = 320  IDORT  =  18
                              IT =  320
WR   = -0.1000E+01  0.1000E+01  0.9935E+00 -0.9935E+00 -0.9755E+00  0.9755E+00
WI   =  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
RSD  =  0.3055E-06  0.1198E-05  0.2302E-05  0.5986E-06  0.1712E-03  0.6930E-03
NGRP =          2                         2                        2
CTR  =       0.1000E+01              0.9935E+00               0.9755E+00
AE   =      -0.2980E-07              0.2980E-06               0.0000E+00
ARSD =       0.8745E-06              0.1682E-05               0.5047E-03
```

The course of the iteration is unexceptionable. The program doubles the interval between SRR step 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 actually costs, runs were made with $m = 2, 4, 6, 8$, which gave the following table of iterations and timings (in second) for the convergence of the first group of two eigenvalues.

m	it	$m \times it$	run time
2	1660	3320	49.84
4	600	2400	37.99
6	320	1920	32.82
8	183	1464	27.45

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. We will see this point in the following example 3.

Example 2. 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

$$\begin{aligned} &y'' + \mu^2 y = 0, \\ &y(0) = 0, \\ &y'(0) + \gamma y'(1) = 0, \quad 0 < \gamma < 1 \end{aligned} \tag{9}$$

The eigenvalues of this problem are easily seen to be given by

$$\mu = i \cosh^{-1}(-\gamma^{-1}),$$

which are complex. The following table lists the reciprocals of the first eight eigenvalues for $\gamma = 0.01$.

$$\begin{array}{cc} \mu^{-2} & |\mu^{-2}| \\ & \\ 0.012644 \pm 0.02313i & 0.02636 \\ 0.004446 \pm 0.00739i & 0.00854 \\ 0.002895 \pm 0.00220i & 0.00364 \\ 0.001274 \pm 0.00089i & 0.00195 \end{array} \tag{10}$$

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

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

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 $B = h^2\mathrm{diag}(1, 1, \ldots, 1, 0)$. 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 $n = 300$ with `M = 6`, `NV = 4`, and `EPS =` 10^{-5} . The results were the following:

```
                              IT =    0
WR   =  0.5990E-02 -0.7362E-03 -0.4792E-03 -0.1994E-03 -0.1419E-03 -0.6238E-04
WI   =  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00
RSD  =  0.2616E-01  0.6177E-02  0.4108E-02  0.1956E-01  0.6401E-02  0.9908E-02
NGRP =    1
CTR  =  0.5990E-02
AE   =  0.5990E-02
ARSD =  0.2616E-01
NXTSRR =   5  IDORT  =   1
                              IT =    5
WR   =  0.1264E-01  0.1264E-01 -0.4476E-02 -0.4476E-02 -0.2732E-02 -0.2732E-02
WI   =  0.2313E-01 -0.2313E-01  0.7324E-02 -0.7324E-02  0.1156E-02 -0.1156E-02
RSD  =  0.1804E-06  0.1804E-06  0.1965E-04  0.1965E-04  0.8603E-03  0.8603E-03
NGRP =    2
CTR  =  0.2636E-01
AE   =  0.1264E-01
ARSD =  0.1804E-06
NXTSRR =  10  IDORT  =   1
                              IT =   10
WR   =  0.1264E-01  0.1264E-01 -0.4447E-02 -0.4447E-02 -0.2838E-02 -0.2838E-02
WI   =  0.2312E-01 -0.2312E-01  0.7308E-02 -0.7308E-02  0.2131E-02 -0.2131E-02
RSD  =  0.2184E-07  0.2184E-07  0.7119E-07  0.7119E-07  0.1584E-03  0.1584E-03
NGRP =        2                      2                      2
CTR  =      0.2636E-01             0.8555E-02             0.3549E-02
AE   =      0.1264E-01            -0.4447E-02            -0.2838E-02
ARSD =      0.2184E-07             0.7119E-07             0.1584E-03
```

Given the extremely favorable ratios of the eigenvalues in table (10) - 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 machinery of *SRRIT*. However, it is an excellent example 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 if the storage available, not the time required to compute Ax. The next example from partial differential equation demonstrates this point again.

Example 3. Let us consider the following sample convection-diffusion problem:

$$\begin{aligned} -\Delta u + 2p_1 u_x + 2p_2 u_y - p_3 u &= 0 \quad \text{in} \quad \Omega \\ u &= 0 \quad \text{on} \quad \partial\Omega \end{aligned}$$

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

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

with

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

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

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

The following lists the first ten eigenvalues for $p_1 = p_2 = p_3 = 1$:

$$\begin{array}{c} 0.7977818E+01 \\ 0.7949033E+01 \\ 0.7949033E+01 \\ 0.7920248E+01 \\ 0.7901366E+01 \\ 0.7901366E+01 \\ 0.7872581E+01 \\ 0.7872581E+01 \\ 0.7835278E+01 \\ 0.7835278E+01 \end{array}$$

The algorithm was run on the 961×961 matrix A obtained by taking 31×31 mesh grid. We are interested in the first dominant eigenvalues. The results obtained are listed in the following table for different value of m (`EPS =` 10^{-4}):

m	λ_{m+1}/λ_1	it	$m \times it$	run time
2	0.9964	1280	2560	18.13
4	0.9904	593	2372	17.55
6	0.9868	320	1920	15.36
8	0.9821	320	2560	21.21

This is a cluster eigenvalue problem, the ratios of the eigenvalues is very closed. As the increase of m, the iteration steps was reduced. However, the total number of matrix-vector multiplications are increased.

Appendix A. List of Subroutines Called by SRRIT

`ATQ` supplied by user, but the calling sequence has to be as described in Section 2.

`SRRSTP` performs an Schur-Rayleigh-Ritz iteration step.

`ORTH` orthonormalizes columns of a matrix.

`RESID` computes the each column norm 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 eigenvalues of Schur form appear in descending order of magnitude along its diagonal. This subroutine is a variant of LAPACK subroutine `SLAHQR` for computing the Schur decomposition.

`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.

`SORGN2` forms all or part of a real orthogonal matrix Q, which is defined as a product of k Householder transformations.

`SLAEQU` Standardization of a 2 by 2 block

Subroutins from BLAS

`ISAMAX` finds the index of element having max. absolute value.

`SCOPY` copy a vector x to vector y.

`SDOT` inner product of two vectors $x^{\mathrm{T}}y$.

`SROT` applies a plane rotation.

`SAXPY` saxpy operation: $\alpha x + y \rightarrow y$.

`SSCAL` scale a vector by a constant.

`SSWAP` interchanges two vectors.

`SNRM2` compute 2-norm of a vector.

`SGEMV` matrix-vector multiplication.

`SGER` performs thr rank 1 updating: $\alpha x \cdot y^{\mathrm{T}} + A \rightarrow A$.

`SGEMM` matrix-matrix multiplication.

Subroutines from LAPACK

`SGEHD2` reduces a full matrix to upper Hessenberg matrix (BLAS 2 code).

`STREXC` moves a given 1 by 1 or 2 by 2 diagonal block of a real Schur matrix to the specified position.

`SLAEXC` swaps adjacent diagonal blocks (1 by 1 or 2 by 2) of a Schur matrix.

`SLARFG` generates Householder transformation.

`SLARF(X)` applies Householder transformation.

`SLASY2` solves up to 2 by 2 Sylvester equation $AX - XB = C$.

`SLALN2` solves up to 2 by 2 linear system equation $(A - \sigma I)x = b$.

`SLANV2` computes the Schur decomposition of a 2 by 2 matrix.

`SLADIV` computes complex division in real arithmetic.

`SLAPY2` computes $\sqrt{a^2 + b^2}$.

`SLARTG` generates a plane rotation.

`SLANGE` computes norm of a general matrix.

`SLANHS` computes norm of a Hessenberg matrix.

`SLASSQ` called by `SLANGE` and `SLANHS`.

`SLAZRO` initializes a matrix.

`SLACPY` copy from one array to another array.

`SLAMCH` determines machine parameters, such as machine precisioni `SLABAD`.

`LSAME` checks character parameter.

`XERBLA` An error handler routine (return error messages).

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. Mckenney, S. Ostrouchov and D. Sorensen, LAPACK Users' Guide, Release 1.0, SIAM, Philadelphia, 1992.

[2] Z. Bai and J. Demmel, On swapping diagonal blocks in real Schur form, submitted to Lin. Alg. Appl. 1992

[3] F. L. Bauer, Das Verfahren der Treppeniteration und verwandte Verfahren zur Losung algebraischer Eigenwertprobleme, Z. Angew, Math. Phys. 8(1957), pp.214-235.

[4] M. Clint and A. Jennings, The evaluation of eigenvalues and eigenvectors of a real symmetric matrix by simultaneous iteration, Comput. J. 13(1970), pp.68-80

[5] A. Jennings and W. J. Stewart, A simultaneous iteration method for the unsymmetric eigenvalue problem, J. Inst. Math. Appl. 8(1971), pp.111-121.

[6] W. Feller, An introduction to probability theory and its applications, John Wiley, New York, 1961

[7] H. Rutishauser, Computational aspects of F. L. Bauer's simultaneous iteration method, Numer. Math. 13(1969), pp.4-13.

[8] H. Rutishauser, Simultaneous iteration method for symmetric matrices, Numer. Math. 16(1970), pp.205-223.

[9] B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema and C. B. Moler, Matrix eigensystem routines – EISPACK guide, Lec. Notes in Comp. Sci. 6, Springer, New York, 1974

[10] G. W. Stewart, Accelerating the orthogonal iteration for the eigenvalues of a Hermitian matrix, Numer. Math. 13(1969), pp.362-376.

[11] G. W. Stewart, Introduction to Matrix Computations, Academic Press, New York, 1973.

[12] G. W. Stewart, Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices, Numer. Math. 25(1976), pp.123-126.

[13] G. W. Stewart, HQR3 and EXCHNG: FORTRAN subroutines for calculating the eigenvalues of a real upper Hessenberg matrix in a prescribed order, ACM Trans. Math. Software 2(1976), pp.275-280.