February 1975 LRP 90/75

H Y M N I A

BAND MATRIX PACKAGE FOR SOLVING EIGENVALUE PROBLEMS

R. Gruber

Centre de Recherches en Physique des Plasmas
ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE

MATHEMATICAL PROBLEM

Stability problems in physics or engineering such as those solved in [1,2,3] lead to eigenvalue problems of the type $A\vec{x} = \lambda B\vec{x}$, where B is positive definite and A can either be real symmetric or hermitian. Both matrices have a band structure.

PROGRAM DESCRIPTION

HYMNIA is a band matrix package which solves the above-mentioned eigenvalue problem by the method of simultaneous inverse vector iteration [4]. Only the eigenvalues with lowest absolute value are obtained directly, but with an eigenvalue shift λ_o it is possible to get any of them.

The advantage of HYMNIA, which is also the name of the CDC main program that tests the two main subprograms* <SIVI> and <CSIVI>, is that all operations are performed within the half band width m of the one-dimensionally stored matrices A and B of order n. By decomposing the matrices ($\sim n \cdot m^2$ operations) before the iteration, only a number of point operations proportional to $n \cdot m$ has to be performed for the iteration part. The NEG facility gives the number of negative eigenvalues for the shifted problem and thus enables us to know which eigenvalue has been calculated.

<SIVI> not only solves the eigenvalue problem, but can also be used simply to decompose A and/or B (Section 2.2). The cases of singular A and non positive B are detected and diagnosed. Convergence information is given by the parameters NCONV and CONV. Typical running times are given in Table 4.

---

*References to subprogram names are in angular brackets < >. Those with prefix "C" refer to the hermitian case and use complex arithmetic.

UNUSUAL FEATURES

The test program HYMNIA reads input data using NAMELIST, which is not STANDARD FORTRAN but is available on most computers.

Another non-standard utility is the CDC random generator RANF called by the subprograms <SIVI>, <CSIVI>, <ORNOS> and <CORNOS>.

REFERENCES

[1] T. Takeda, Y. Shimomura, M. Ohta, M. Yoshikawa
Phys,Fluids 15, 2193 (1972)

[2] K. Appert, D. Berger, R. Gruber, J. Rappaz, LRP 83/74
EPF-Lausanne, CRPP, submitted to J.Comp.Phy,

[3] K. Appert, D. Berger, R. Gruber, K.V. Roberts
Computer Phys.Comm. (previous paper of this issue)

[4] J.H. Wilkinson, The Algebric Eigenvalue Problem,
Clarendon Press, Oxford, 1965

1. INTRODUCTION

<HYMNIA> is a main program which tests a band matrix package. The package solves the eigenvalue problem

$$A\vec{x} = \lambda B\vec{x} \tag{1}$$

by the method of simultaneous inverse vector iteration [1,2,3,4] <SIVI>. The matrix B has to be positive definite and real, and A must be either real symmetric <SIVI> or hermitian <CSIVI>. An inverse vector-iteration converges to the lowest eigenvalue λ in absolute value and its corresponding eigenvector $\vec{x}$. In order to obtain r eigenvalues lying near λ_o and their corresponding eigenvectors, we perform an eigenvalue shift of problem (1)

$$\tilde{A}\vec{x} = \tilde{\lambda} B\vec{x} \tag{2}$$

where $\tilde{A} = A - \lambda_o B$ and $\tilde{\lambda} = \lambda - \lambda_o$. When the eigenvalues are found by doing such shifts sequentially one speaks of a Sturm sequence or a bisection method. Such a method has been used in [5] but takes too much computing time. A better method is to solve (2) by an inverse vector iteration which takes into account the Sturm sequence [4] by the parameter NEG, i.e. the number of eigenvalues less than λ_o. The different ways of using <SIVI> are given in Table 1.

HYMNIA has been designed as an eigenvalue problem-solving package for MHD stability problems. The one-dimensional MHD stability code THALIA which calls <SIVI> is described in the previous paper [6], while the two-dimensional case which uses <CSIVI> is under construction. HYMNIA will later be extended to include non-symmetric eigenvalue problems, e.g. for studying resistive instabilities in a plasma.

In section 2 we describe the mathematical method for solving (1), in sec-

tion 3 the numerical solution with tests and timing information, while section 4 contains some useful instructions for the user. The appendix gives a list of subprograms and the input/output parameters of <SIVI>.

Note that in the mathematical descriptions we only treat the hermitian case, which of course includes the real one. Cross-references to the real subprograms are given; the corresponding subprogram for the complex case contains a C preceding the subprogram name.

Table 1: Cases for which <SIVI> can be used

Case	NTYPE	
1	0	Solve eigenvalue problem $(A - \lambda B)\vec{x} = 0$
2	1	Decompose $A = L\,DL^H$ and $B = R^TR$
3	2	Decompose $A = L\,DL^H$
4	3	Decompose $B = R^TR$

2. MATHEMATICAL PROBLEM

2.1 Shift of the eigenvalue problem

To find all the eigenvalues a shift of problem (1) has to be performed <SHIFT>. We then solve (2) by an inverse vector iteration [1,2,3,4]. The choise of shift λ_o has to be done carefully, as follows from a simple convergence consideration:

Let λ_1 and λ_2 be the two eigenvalues nearest to λ_o with $|\lambda_1 - \lambda_o| < |\lambda_2 - \lambda_o|$.

Then the convergence rate is given by $\omega = |\lambda_1 - \lambda_o| / |\lambda_2 - \lambda_o|$ [7]. λ_o has to be chosen by the user a priori in such a way that ω becomes small.

2.2 Decomposition of A and B

2.2.1 Decomposition of A

Let A be a regular hermitian band matrix. Whenever all principal submatrices of A are regular, a unique decomposition

$$A = L D L^H \tag{3}$$

can be performed into a left hand side triangular band matrix L and its transpose and conjugate L^H containing unities on the diagonal, together with a diagonal matrix D.

This is rapidly checked by counting the independent matrix elements on each side. An example of such a decomposition is given in section 3.2.1.

2.2.2 Decomposition of B

Let B a real symmetric and definite positive band matrix. A unique, so-called Cholewski-decomposition

$$B = R^T R \tag{4}$$

can be performed into a right hand side triangular band matrix R and its transpose R^T. Note that (4) is a special case of the more general decomposition (3).

2.2.3 Rounding errors

It is well known [8] that elimination processes such as (3) and (4) give

rise to accuracy destruction which is proportional to the number of point operations in the elimination. In the band matrix case the losses of numerical accuracy in a decomposition are proportional to $n \cdot m^2$, where n is the matrix length and m the band width. The proportionality factor can be very high if the pivot for the elimination is badly chosen, but in order to preserve the band structure of our matrices there is no possibility of choosing the pivot. This disadvantage is not too severe here, however, because in most of the physical problems to be studied the matrices are diagonal dominant.

Whenever decomposition errors are too high it is always possible to introduce in the code a backward analysis of the Wilkinson type [7,8].

2.2.4 Discussion

In order to solve (1) one solves another eigenvalue problem

$$L D L^H \vec{x} = \lambda R^T R \vec{x} . \tag{6}$$

For a simultaneous iteration it is necessary to know which vectors have to be orthogonal, that is one has to construct an eigenvalue problem

$$A_s \vec{x} = \lambda I \vec{x} \tag{7}$$

where I is the identity and the u's are orthogonal for a hermitian matrix A_s.

Let us construct (7) by starting from (1). Using the decomposition (4), (1) can be written as

$$A \vec{x} = \lambda R^T R \vec{x} \tag{8}$$

Introducing a new eigenvector

$$\vec{u} = R \vec{x} \tag{9}$$

and multiplying (8) with $(R^{-1})^T$ from the left one obtains just (7) with

$$A_s = (R^{-1})^T A R^{-1} . \tag{10}$$

The new problem (7) has a big disadvantage: A is a full hermitian matrix, coming from the fact that the triangular inverted matrix R^{-1} is also full. Nevertheless we have learned something. We know now that we have to orthogonalize the vectors $\vec{u} = R\vec{x}$.

2.3 Number of negative eigenvalues

After the decomposition (3) we know how many eigenvalues are less then the shift λ_o. This follows from Theorem 1:

Theorem 1: The number of negative eigenvalues of the problem (1) is equal to the number of negative terms of the diagonal matrix D in the decomposition (3).

Proof : To prove Theorem 1 we use Sylvester's inertia theorem [9]:
"The number of negative eigenvalues of a Hermitian matrix A is unchanged after any transformation TAT^H, where T is regular".

Theorem 1 follows at once. With (3) and (10) A, D and A_s have the same number of negative eigenvalues. It is now evident that D contains the same number of negative eigenvalues as problem (1).

The possibility of knowing the number of negative eigenvalues allows us never to miss one. Whenever, for example, the number NEG of negative eigenvalues is 0 we know that the given converged eigenvalue is really the smallest one. With this NEG facility we could use <SIVI> to find the eigenvalues by a bisection method, i.e. by choosing e.g. two different shifts λ_1 and λ_2 one knows how many eigenvalues lie in between ($|$NEG2 - NEG1$|$). By cutting the eigenvalue-intervals into two consecutively, one could isolate one eigen-

value after the other (Sturm sequence). This method is however not recommended because it is time consuming: The number of time steps is proportional to the number of interval-cuts, to the matrix length and to the square of the band width.

2.4 Starting vectors

The iteration process

$$L D L^H \vec{x}_{k+1} = R^T R \vec{x}_k \tag{11}$$

where the r vectors $\vec{u}_k = R\vec{x}_k$ (9) are orthonormalized by <ORNOS> , must be started from a set of initial vectors $\vec{x}_o$. These initial vectors have to be linearly independent, and in <SIVI> we create them using a random-number generator.

2.5 Iteration for Eigenvectors

2.5.1 Iterative process

To describe the iteration we define some new vectors:

$$\vec{u}_k = R\vec{x}_k \tag{12}$$

$$\vec{v}_k = R^T\vec{u}_k \tag{13}$$

$$\vec{y}_{k+1} = DL^H\vec{x}_{k+1} \tag{14}$$

$$\vec{w}_{k+1} = L^H\vec{x}_{k+1} \tag{15}$$

The iteration process (11) is then performed in the following way:

1. Multiply $\vec{u}_K = R\vec{x}_K$ (16)

2. Orthonormalize $\vec{u}_K$ (17)

3. Multiply $\vec{v}_K = R^T\vec{u}_K$ (18)

4. Solve $L\vec{y}_{k+1} = \vec{v}_K$ (19)

5. Solve $D\vec{w}_{k+1} = \vec{y}_{k+1}$ (20)

6. Solve $L^H\vec{x}_{k+1} = \vec{w}_{k+1}$ (21)

2.5.2 Iteration stop criterion

The iteration stop criterion is applied to all components i of the orthonormalized vectors $\vec{u}$ (9). Iteration stops when

$$\left| u_{k+1}^{(i)} - \sigma u_k^{(i)} \right| < \varepsilon \qquad \forall i = 1 \ldots, rn, \tag{22}$$

where σ = sign $(\lambda - \lambda_o)$. In <SIVI>, we do not know a priori if $\lambda - \lambda_o$ is positive or negative. We find σ by working out the sign of the biggest component of $\vec{u}_{k+1}$. In the code ε is denoted by EPSCON.

2.6 Eigenvalues

The iteration process (16-21) furnishes the r eigenvectors $\vec{x}$. To find the eigenvalues one multiplies (1) by a general test vector $\vec{z}^H$ and divides:

$$\lambda = \frac{\vec{z}^H A\vec{x}}{\vec{z}^H B\vec{x}} \quad . \tag{23}$$

(23) is well known as the generalized Rayleigh Quotient [1]. The best choice for $\vec{z}$ is the eigenvector $\vec{x}$ itself: $\vec{x}^H B\vec{x}$ is then always positive.

Sometimes (when the eigenvectors did not converge) it is convenient to choose $\vec{z} = \vec{e}_i$, i.e. unit vectors for all components. The r "eigenvalue vectors" which are obtained then contain a lot of information about the location of the eigenvalues, because a non-converged eigenvector consists of a mixture of neighbouring modes.

2.7 Timing

2.7.1 Timing for decompositions

Decompositions (3) and (4) need a number of point operations proportional to $n \cdot m^2$. Let t_1 be the time to perform a point operation for (3) and (4) together. The total computing time T_1 for both decompositions is then

$$T_1 = n \cdot m^2 \cdot t_1 \tag{24}$$

t_1 will be found empirically in section 3.5.

2.7.2 Timing for iteration

The matrix multiplications (16) and (18), the orthonormalization (17) and the backward substitutions (19), (20) and (21) all take an amount of computer time proportional to n * m * NKV * NIT, where NKV = r is the number of iterations used to find them. Let t_2 be the time used to perform one point operation for each step. The total computing time T_2 for the iteration is then given by

$$T_2 = n \cdot m \cdot NKV \cdot NIT \cdot t_2 \tag{25}$$

t_2 also will be found empirically in section 3.5.

2.7.3 Discussion

The total computing time T is given by

$$T = T_1 + T_2 = n \cdot m \cdot (m \cdot t_1 + NKV \cdot NIT \cdot t_2) \tag{26}$$

We shall see in section 3.5 that t_2 is about 4 times bigger than t_1. When the band width m is not too high the second term in (26) is mostly bigger than the first one. In this case it is not convenient to iterate on more than one eigenvalue at once, because one increases the number of iterations NIT by increasing NKV. Simultaneous iterations on several eigenvectors have to be done for degenerate cases where one has to construct a subspace for the eigenvectors.

3. NUMERICAL SOLUTION

3.1 Notation

Fig. 1 shows on the left-hand side the usual notation for an element a(i,j) of a full hermitian matrix A. In the middle a twice subscripted half band matrix is presented. Similar elements in the twice subscripted matrix and in the full hermitian matrix have the correspondence

$$\begin{aligned} i_b &= i \\ j_b &= j - i + 1 \end{aligned} \tag{27}$$

where b denotes band.

In <SIVI> the half band matrix is one-dimensionally subscripted by k as shown in the right hand side of Fig. 1. The matrix is stored row by row. The correspondences to the full and the twice subscripted matrix are

$$k = (i-1)\cdot(m-1)+j$$
$$k = (i_b-1)\cdot m + j_b \cdot \qquad (28)$$

One-dimensional notation saves time in general because it is usually possible to replace the address calculation (multiplication + addition for a twice subscripted array) by a simple addition. Some notational problems rise when we have to handle a whole row or a whole column of a full hermitian matrix in a half band. Fig. 2 shows how a row of a full matrix (upper left) is represented in a band matrix (upper right). The left hand side part (always relative to the diagonal) is stored as its complex conjugate ascending from the diagonal, whereas the right hand side part follows (28).

Similar problems have to solved when we want to treat a column. The upper full matrix part corresponds to the ascending elements and the lower part to the complex conjugate of the row (Fig. 2).

3.2 Decompositions

3.2.1 Decomposition of A

We explain the decomposition (3) of a real $A = LDL^T$ by means of an example. Let us choose a 4 x 4 matrix A. The problem is to calculate the l_{ij}'s and d_k's of the matrices L and D respectively.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ 0 & a_{24} & a_{34} & a_{44} \end{pmatrix} \qquad (29)$$

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{12} & 1 & 0 & 0 \\ l_{13} & l_{23} & 1 & 0 \\ 0 & l_{24} & l_{34} & 1 \end{pmatrix} \tag{30}$$

$$D = \begin{pmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{pmatrix} \tag{31}$$

$$LDL^T = \begin{pmatrix} d_1 & l_{12}d_1 & l_{13}d_1 & 0 \\ l_{12}d_1 & l_{12}^2d_1+d_2 & l_{12}l_{13}d_1+l_{23}d_2 & l_{24}d_2 \\ l_{13}d_1 & l_{13}l_{12}d_1+l_{23}d_2 & l_{13}^2d_1+l_{23}^2d_2+d_3 & l_{23}l_{24}d_2+l_{34}d_3 \\ 0 & l_{24}d_2 & l_{24}l_{23}d_2+l_{34}d_3 & l_{24}^2d_2+l_{34}^2d_3+d_4 \end{pmatrix} \tag{32}$$

A (29) and LDL^T (32) now have to be identified. The rule of calculation is presented in Table 2 (only the upper symmetric part is treated).

Table 2: Decomposition of A

Variable definition	Stored in	Loop Index
1. $d_1 = a_{11}$	a_{11}	k = 1
2. $l_{1\ell} = a_{1\ell}/a_{11}$	$a_{1\ell}$	ℓ = 2,3
3. $a_{22} = a_{22} - a_{12}^2 a_{11}$		
4. $a_{23} = a_{23} - a_{13}a_{12}a_{11}$		
5. $a_{33} = a_{33} - a_{13}^2 a_{11}$		
6. $d_2 = a_{22}$	a_{22}	k = 2
7. $l_{2\ell} = a_{2\ell}/a_{22}$	$a_{2\ell}$	ℓ = 3,4
8. $a_{33} = a_{33} - a_{23}^2 a_{22}$		
9. $a_{34} = a_{34} - a_{24}a_{24}a_{22}$		
10. $a_{44} = a_{44} - a_{24}^2 a_{22}$		
11. $d_3 = a_{33}$	a_{33}	k = 3
12. $l_{3\ell} = a_{3\ell}/a_{33}$	$a_{3\ell}$	ℓ = 4
13. $a_{44} = a_{44} - a_{34}^2 a_{33}$		
14. $d_4 = a_{44}$	a_{44}	k = 4

This algorithm assumes that A admits a Gauss elimination. Whenever this is not the case, i.e. when a pivot becomes 0, an automatic shift of the eigenvalue spectrum as described by (2) is performed. Not that all calculations are performed within the storage area of the band matrix A itself (Fig. 3).

When A is singular, at most one of the pivots vanishes. The subprograms < ALDLT > and <CALDLT > consider a pivot as zero when in its further

treatment by the rectangular rule, for example at operations 3,8 or 13, the absolute value of the diagonal element is destroyed relatively to EPSMAC * N * M^2, the relative machine accuracy which for a CDC 6500 in [6] was set to 10^{-10}. When a principle submatrix of A is found to be singular the variable NSING is put to -1 (NSING = 0 for regular A), and execution returns to < SIVI > or <CSIVI > where the two diagnostic messages

*** PRINCIPAL SUBMATRIX OF A IS SINGULAR ***

*** AN AUTOMATIC SHIFT HAS BEEN PERFORMED *** (33)

are printed out. An additional automatic shift of λ_o = 100*EPSMAC*N*M^2 is then performed and problem (2) is subsequently solved. NSING is reset to zero.

3.2.2 Decomposition of B

The so-called Cholewski-decomposition of B is performed in a similar way to the decomposition of A (example in Table 2). All operations in <BRT> (there does not exist a complex version) are done in the array itself. As discussed in 2.2.2. B has to be real symmetric and definite positive. B not positive gives a message

*** MATRIX B NOT POSITIVE *** (34)

and the eigenvalue problem is not solved.

3.3 Iteration subprograms

The iteration (16) - (21) is performed by the subprograms in Table 3.

Table 3:

Operation	Real subprogram	Complex subprogram
1. $\vec{u}_k = R\,\vec{x}_k$	URX	CURX
2. ON $(\vec{u}_k)$	ORNOS	CORNOS
3. $\vec{v}_k = R^T\,\vec{u}_k$	VRTU	CVRTU
4. L $\vec{y}_{k+1} = \vec{v}_k$	LYV	CLYV
5. D $\vec{w}_{k+1} = \vec{y}_{k+1}$	DWY	CDWY
6. L $\vec{x}_{k+1} = \vec{w}_{k+1}$	LTXW	CLTXW

3.3.1 Orthonormalization

The orthogonalizing process uses Schmidt's orthogonalization method [2]. The second vector is orthogonalized relative to the first one, the third relative to the first and the second one and so on.

For a degenerate case it is necessary to construct a vector subspace in which the vectors are orthogonal. Whenever vectors are linearly dependent we choose another one at random and orthonormalize it afterwards. Testing for dependency is done in a similar way to the test for singular A.

3.3.2 Beckward substitution

Operations (4. - 6.) in Table 3 are simple backward substitutions. In 4. we start the resolution at the top of the vector $\vec{y}_{k+1}$ and go downwards. The number of characteristic operations is proportional to n*m. In 5. the resolution is trivial. Only n operations have to be done. The last backward subsitution 6. starts at the end of vector $\vec{x}_{k+1}$ and solves up to the top of the vector.

3.4 Tests

The main program HYMNIA is a test program for the band matrix package. HYMNIA contains 4 tests:

Test 1 / Test 2 The eigenvalue problem

$$\begin{pmatrix} 3 & 6 & 0 \\ 6 & 10 & -2 \\ 0 & -2 & -2 \end{pmatrix} \vec{x} = \lambda \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \vec{x} \qquad (35)$$

is solved, whose exact solution is

$$\lambda = \left[\frac{1}{2}\left(11-\sqrt{265}\right), 0, \frac{1}{2}\left(11-\sqrt{265}\right)\right] \qquad (36)$$

$$\vec{x}^T(\lambda) = \left[-\frac{6x_2}{3-\lambda}, x_2, -\frac{2x_2}{2+\lambda}\right] \qquad (37)$$

with

$$x_2^2 = \frac{(2+\lambda)^2(3-\lambda)^2}{216-108\lambda+29\lambda^2-2\lambda^3+\lambda^4} \quad \text{and} \quad \|\vec{x}\| = 1 .$$

For the case $\lambda = 0$, (37) becomes

$$\vec{x}^T = \left(-2\sqrt{\frac{1}{6}}, \sqrt{\frac{1}{6}}, -\sqrt{\frac{1}{6}}\right) . \qquad (38)$$

TEST 1 does an eigenvalue shift with $\lambda_o = 10$ in order to obtain the largest eigenvalue $\lambda_3 = 13.6394$.

TEST 2 is given as TEST RUN OUTPUT. The shift value is chosen to be $\lambda_o = 0$ and matrix A becomes singular. An automatic shift with 100·EPSMAC* $n*m^2*B$ is performed afterwards and the shifted problem (2) is solved. The

two simultaneously iterated eigenvalues, lowest in absolute value, $\lambda_1 = 0$ and $\lambda_2 = -2.63941$ are obtained.

TEST 3: Matrix B is not positive in this test.

TEST 4: The complex part of HYMNIA <CSIVI> which treats a hermitian matrix A is tested by the problem

$$\begin{pmatrix} -8 & -4-12i & -12+4i & 0 \\ -4+12i & -5 & 1-12i & 6+3i \\ -12-4i & 1+12i & 12 & -5+11i \\ 0 & 6-3i & -5-11i & 7 \end{pmatrix} \vec{x} = \lambda \begin{pmatrix} 8 & 2 & -1 & 0 \\ 2 & 12 & 4 & -2 \\ -1 & 4 & 11 & 1 \\ 0 & -2 & 1 & 6 \end{pmatrix} \vec{x} \qquad (39)$$

The eigenvalue near $\lambda_o = -3$ and its corresponding eigenvector are

$$\lambda = -2.57752$$

$$\vec{x}^T = (.473-.625i, -.442+.008i, .4-.067i, .091+.129i) \; . \qquad (40)$$

3.5 Timing

Separate timing procedures (26) have been carried out for real and for hermitian A. For real A the timing has been performed with matrices n = 303 and m = 6 (FTN(OPT=1)-Compiler). For hermitian A the matrix dimensions were n = 195 and m = 22 (RUN-Compiler). Table 4 gives the measured computing times for NKV = 1.

Table 4: Timing

Dimension of A, B	Type of A	Decomposition time	t_1	NIT	Iteration time	t_2
n = 303, m = 6	real	0.44 s	40 μs	12	3.45 s	160 μs
n = 195, m = 22	hermitian	4.6 s	50 μs	40	40.7 s	240 μs

For a CDC Cyber 7326 the timing formula (26) becomes for real A

$$T = n*m*(40*m + 160*NKV*NIT) \quad \mu s \tag{41}$$

whereas for hermitian A (26) is given by

$$T = n*m*(50*m + 240*NKV*NIT) \quad \mu s \tag{42}$$

We see that in both cases the iteration time is in general bigger than the "mathematically important" part $n.m^2$ coming from the decompositions.

4. INSTRUCTIONS FOR THE USER

4.1 Choise of EPSCON and EPSMAC

The iteration stop parameter EPSCON should not be chosen too small. In our application [6] we put as default parameter EPSCON = 10^{-4}.

EPSMAC is a computer-dependent parameter. For a 60 bit word machine, for example, the last significant bit has a relative error of about 10^{-13}. In [6] EPSMAC has been chosen to be 10^{-10}.

4.2 Choice of λ_o

The choice of the shift λ_o has to be done carefully. For example in order to find the lowest eigenvalue, (in [6,10] the lowest negative eigenvalue denotes the most unstable mode), one has to know approximately to look for it. If nothing is known about the position of the eigenvalues it is preferable first to try with $\lambda_o = 0$ and only to look at the number of

negative eigenvalues NEG. A negative λ_o will then decrease NEG, and the best choice of λ_o to find the lowest eigenvalue is that which just makes NEG = 0. This method of varying λ_o is well known as a Sturm sequence, for which the number of operations is proportional to $n*m^2*NIT*NKV$.

4.3 Choice of NKV

In most problems it is convenient to choose the number of simultaneous eigenvalues NKV = 1. This follows from the considerations in 2.7.3 and in 3.5, where the timing was discussed. However for a degenerate problem, where a subspace of eigenvectors has to be constructed, a simultaneous iteration over all degenerate modes has to be performed.

If one does wish to iterate on several eigenvalues for a non-degenerate case, the convergence properties become better when one iterates over more eigenvalues than are really required [4].

4.4 Choice of NITMAX

NITMAX, the maximum number of iterations, should not be too high. Increasing NITMAX might make it necessary to increase the DIMENSION declaration of the array CONV (appendix), which shows how the vector displacement converge during the iteration process. More precisely CONV(k) contains the sum of the euclidean norms of the NKV vectors

$$CONV(k) = \sum_{r=1}^{NKV} \| \vec{u}_{k+1} - \vec{u}_k * sign(\lambda - \lambda_o) \|^2 \qquad (43)$$

where k is the iteration counter. NCONV, another output parameter from <SIVI>, contains the number of converged vector-components which will by identical to the total number of components when NIT (= number of required iterations) is less than NITMAX.

4.5 Unusual features of the program

The test program HYMNIA reads the data in by the NAMELIST facility, which is available on most computers. If NAMELIST cannot be used the statement

```
READ (NIN,TEST)                                    (44)
```

should by replaced by an ordinary FORTRAN READ statement which includes all variables listed in namelist TEST.

The subprograms <SIVI> and <ORNOS> call a system routine RANF(N), where N is a dummy argument. RANF is the CDC random number generator. If such a routine of the name is not available the user should introduce a corresponding version of his own.

ACKNOWLEDGMENT

The author wishes to thank J. Rappaz and Dr. K.V. Roberts for a number of discussions and for fruitful comments on the subject of this paper.

This work was supported by the "Fonds National Suisse de la Recherche Scientifique".

APPENDIX

HYMNIA : Subprogram Names and UPDATE identifiers

NAME	UPDATE identifier	Number of arguments	TITLE
SIVI	MR01	10	Solve $A*\vec{x} = \lambda*B*\vec{x}$
SHIFT	MR02	4	Shift by λ_o
ALDLT	MR03	5	Decompose $A = L*D*L^T$
BRTR	MR04	4	Decompose $B = R*R^T$
URX	MR05	6	Multiply $\vec{u} = R*\vec{x}$
ORNOS	MR06	4	Orthonormalize $\vec{u}$
VRTU	MR07	6	Multiply $\vec{v} = R^T*\vec{u}$
LYV	MR08	5	Solve $L*\vec{y} = \vec{v}$
DWY	MR09	5	Solve $D*\vec{w} = \vec{y}$
LTXW	MR10	5	Solve $L^T*\vec{x} = \vec{w}$
UBX	MR11	6	Multiply $\vec{u} = B*\vec{x}$
NORM	MR12	3	Normalize $\vec{x}$
CSIVI	MC01	10	Solve $A*\vec{x} = \lambda*B*\vec{x}$
CSHIFT	MC02	4	Shift by λ_o
CALDLT	MC03	5	Decompose $A = L*D*L^H$
CURX	MC05	6	Multiply $\vec{u} = R*\vec{x}$
CORNOS	MC06	4	Orthonormalize $\vec{u}$
CVRTU	MC07	6	Multiply $\vec{v} = R^T*\vec{u}$
CLYV	MC08	5	Solve $L*\vec{y} = \vec{v}$
CDWY	MC09	5	Solve $D*\vec{w} = \vec{y}$
CLTXW	MC10	5	Solve $L^H*\vec{x} = \vec{w}$
CUBX	MC11	6	Multiply $\vec{u} = B*\vec{x}$
CNORM	MC12	3	Normalize $\vec{x}$

SIVI : Argument list

Argument	[Dimension]	TYPE	Input (I) or output (O)	Title
A	[N * M]	RA	I/O	Matrix A of problem (1)
B	[N * M]	RA	I/O	Matrix B of problem (1)
X	[NKV * N]	RA	O	NKV eigenvectors
U	[NKV * N]	RA	O	"Eigenvalue" vector
V	[NKV * N]	RA		Storage area
AL	[NKV]	RA	O	NKV eigenvectors
AL (1)		R	I	Shift λ_o
CONV	[NITMAX]	RA	O	Convergence properties
EPSCON		R	I	Convergence rate for $\vec{x}$
EPSMAC		R	I	Computer accuracy
NPIN	[7]	IA	I	Input quantities:
NPIN (1) = 0		I	I	Vector iteration
NPIN (1) = 1		I	I	Decomposition with shift
NPIN (1) = 2		I	I	Decomposition of A
NPIN (1) = 3		I	I	Decomposition of B
NPIN (2) = N		I	I	Matrix length
NPIN (3) = M		I	I	Half band width
NPIN (4) = NKV		I	I	Number of iterated modes
NPIN (5) = NITMAX		I	I	Max. number of iterations
NPIN (6) = NSAVE		I	I	Channel number of saving matrix
NPIN (7) = NOUT		I	I	Output channel number
NPIN	[5]	IA	O	Output quantities:
NPOUT (1) = NEG		I	O	Number of $\lambda < \lambda_o$
NPOUT (2) = NIT		I	O	Number of iterations
NPOUT (3) = NCONV		I	O	Number of converged components components
NPOUT (4) = NPOS=0		I	O	B is positive
NPOUT (4) = NPOS=-1		I	O	B is not positive
NPOUT (5) = NSING=0		I	O	A is regular
NPOUT (5) = NSING=-1		I	O	A is singular

REFERENCES

[1] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965

[2] H.R. Schwarz, Numerik symmetrischer Matrizen, B.G. Teubner, Stuttgart, 1972

[3] R.P. Tewarson, Sparse Matrices, Academic Press, N.Y., 1973

[4] K.J. Bathe, E.L. Wilson, Int.J.Numm.Meth.Engng 6, 213-226 (1973)

[5] K.K. Gupta, Int.J.Numm.Meth.Engng 4, 379-404 (1972)

[6] K. Appert, D. Berger, R. Gruber, K.V. Roberts
Computer Phys.Commun. (previous paper of this issue)

[7] H. Werner, Prahtische Mathematik I, Springer-Verlag, Berlin (1970)

[8] J.H. Wilkinson, Rounding errors in algebraic processes,
Her Majesty's Stationery Office, 1963

[9] W.H. Greub, Linear Albegra, Springer-Verlag, Heidelberg, 1967

[10] K. Appert, D. Berger, R. Gruber, J. Rappaz, LRP 83/74, EPF-Lausanne, CRPP, submitted to J.Comp.Phys.

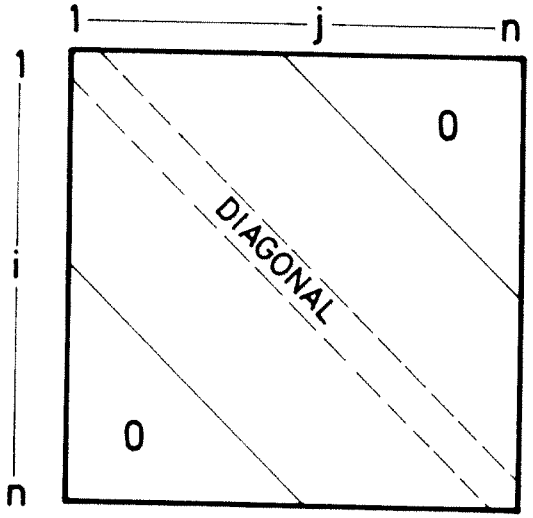
FIGURE CAPTIONS

Fig. 1: Matrix notations

Fig. 2: Storage problems (H denotes the transposed and complex conjugate)

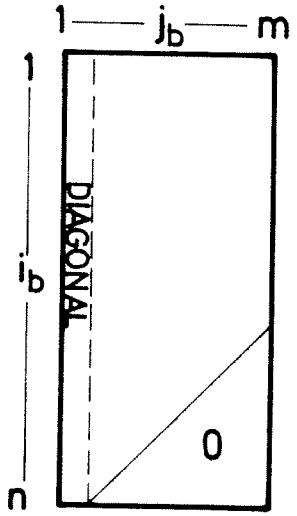
Fig. 3: Decompositions of A and B are performed within their storage areas.

Hermitian matrix A



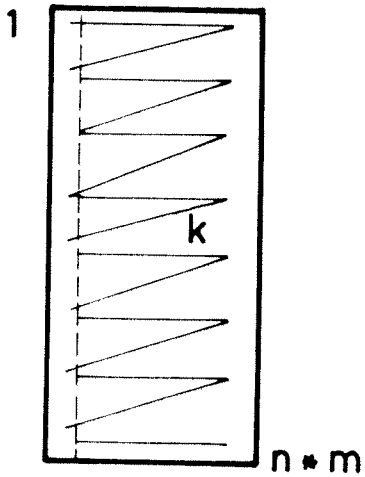
a (i, j)

Twice subscripted
half-band matrix A



a (i_b, j_b)

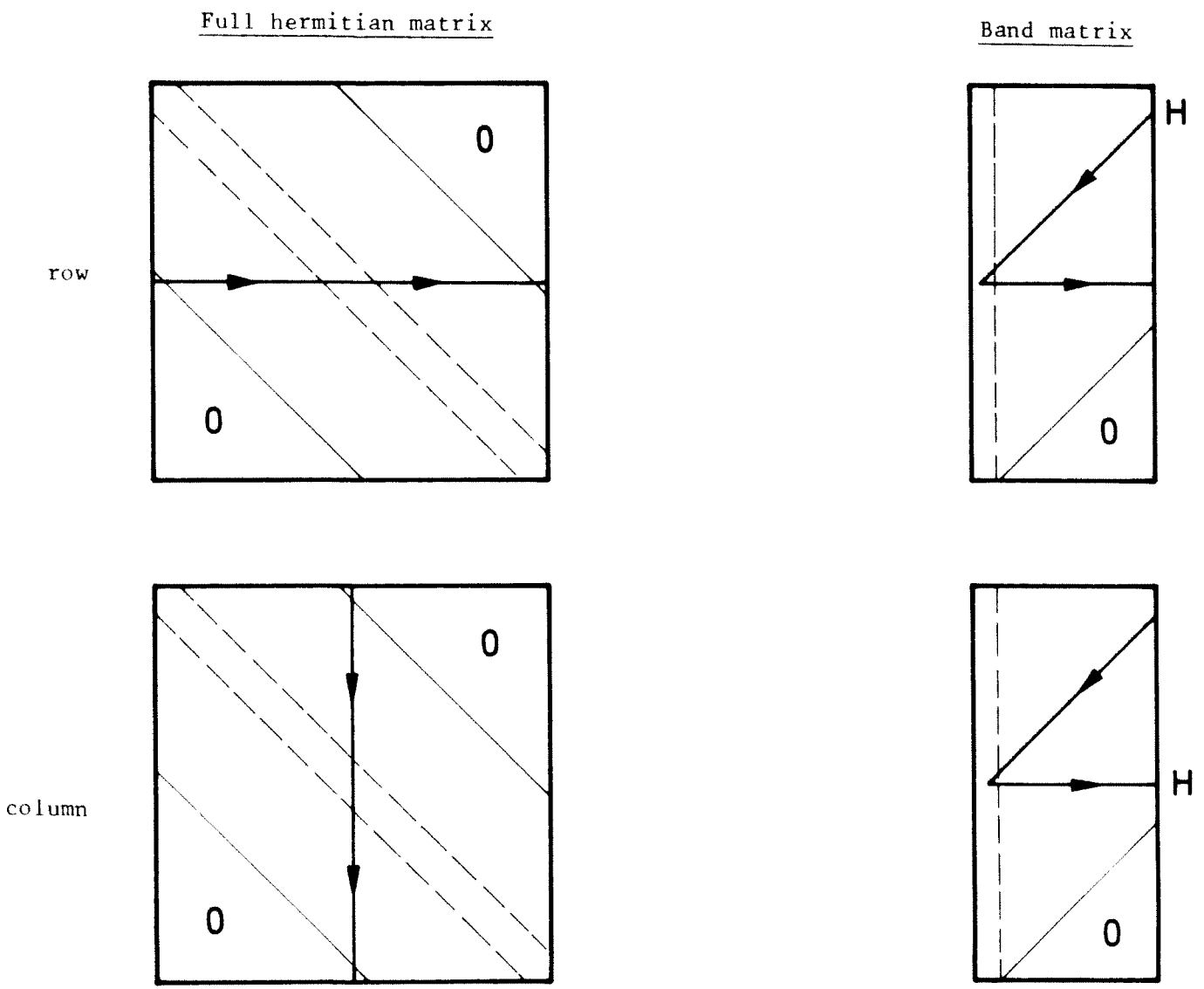
One dimensional
half-band matrix A
as used in <SIVI>

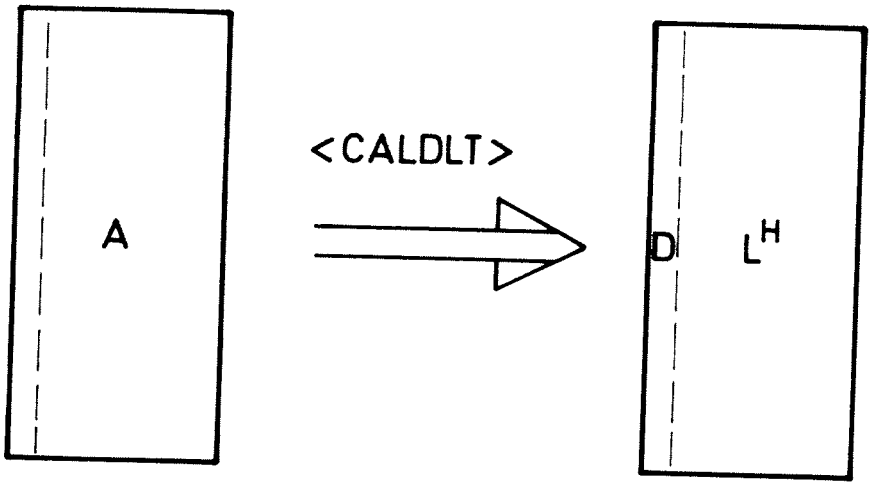


a (k)

Fig. 1

Fig. 2





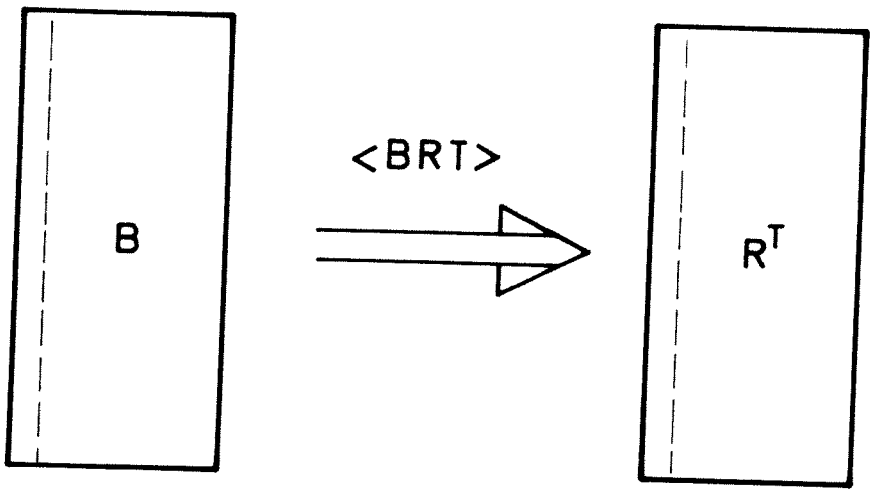


Fig. 3