MARCH 1981 LRP 181/81

Implementing Sparse Matrix Techniques in the ERATO Code

D.S. Scott*, and R. Gruber

* Oak Ridge National Laboratory, Oak Ridge, TN 31830

Implementing Sparse Matrix Techniques in the ERATO Code

D.S. Scott
Oak Ridge National Laboratory, Oak Ridge, TN 31830, USA

R. Gruber
Centre de Recherches en Physique des Plasmas
Association Confédération Suisse - Euratom
Ecole Polytechnique Fédérale de Lausanne
CH-1007 Lausanne / Switzerland

ABSTRACT

In the variational form of the ideal magnetohydrodynamic equations for a toroidal geometry two of the three displacement components appear to be "one dimensional". By an adequate discretization procedure the matrix blocks of the block diagonal matrix do not entirely overlap. This feature can be used to advantage when solving the eigenvalue problem by sparse matrix techniques. Appreciable gain in computing time, memory and disk storage as well as in input-output operations can be obtained.

1. Introduction

The ERATO /1/ code computes the stability of an equilibrium with respect to the linear magnetohydrodynamic equations in an axisymmetric geometry. This code has input - output problems /2/ when run on a very rapid computer such as CRAY 1, CDC Cyber 203, Fujitsu M200 or an IBM 3033 AP. The turn-around time can be 2 to 5 times the CPU time. Big cases, say more than 60 intervals in both directions, can sometimes cause disk overflow. Running ERATO on slow machines such as the VAX or CYBER 170-720, CPU time and, for the CYBER, memory space limit the number of intervals to less than 40 x 40. These difficulties come from the part which solves the eigenvalue problem $A\underline{x} = \omega^2 B\underline{x}$ by inverse vector iteration using the HYMNIABLOCK /3/ package. In this package the sparseness inside the blocks of the block diagonal matrices A and B is not taken into account.

The first attempt to diminish CPU time, disk and memory storage and number of input - output operations was to consider only the stability index /4/, i.e. only the sign of the potential energy matrix A was checked. In this δW code we got rid of one component by the incompressibility condition. A second component appeared to be blockwise one-dimensional and could be eliminated easily.

In the present paper we show for the full problem how we can profit from the blockwise "one-dimensionality" of two of the three displacement components obtaining the same gains as in /4/. The idea is to renumber the unknowns in each block of A so that the non-overlapping part in each matrix block remains untouched when decomposing the matrix A and reorder

the subblocks in between each matrix block of A. We also take full advantage of the sparseness of the matrix blocks of B when we multiply the eigenvector by B at the beginning of each iteration. In the present version the gain in CPU time and memory storage compared with that of the published version /1/ for a case with 50 intervals in both directions is roughly a factor of 4. The amount of input - output operations as well as the total disk space used is reduced by one order of magnitude.

2. The Numerical Problem

The eigenvalue solver in ERATO code has to find the eigenvalue ω^2 and the corresponding eigenfunction $\underline{x}$ of the eigenvalue problem

$$A\underline{x} = \omega^2 B\underline{x} . \qquad (1)$$

The two matrices A and $\omega^2 B$ represent the discretized potential and kinetic energies, respectively. They are both symmetric and B is positive definite. The structure of the matrices is shown in Figure 1. They are both block diagonal. Each block is subdivided into 16 subblocks (see Figure 2). Each subblock consists of a band matrix with a band width of 7. The overlap with the previous block is made through the first subblock (1,1). The overlap with the next block is made through the last subblock (4,4). This matrix structure has been obtained by discretizing the ideal MHD equations /5/ by a finite hybrid element approach /1/. The non-overlapping midparts of the blocks come from the fact that there are no radial derivatives in two components (V,Y) of the displacement vector $\underline{x}$ = (X,V,Y). The discretization is chosen such that V and Y are piecewise constant in the radial s-direction. The position of the nodal variables has been chosen as shown in Figure 3. At the mesh points we

place the component X, the V and the Y nodal points are situated at the center of each s-interval. Since there is no interaction in the radial direction between the V and the Y components, we can consider them to be one-dimensional for each s-interval. This one-dimensionality is lost when we perform the decomposition of the matrix A without touching the structure shown in Figures 1 and 2. This is the case when we solve the eigenvalue problem (Eq. 1) by the block diagonal matrix solver HYMNIA-BLOCK /3/. Only the upper triangles of zeros in the subblocks (1,2), (1,3) and (1,4) can be used to diminish the computing time by about a factor of 2. We show now how we can make maximum use of the sparseness of each matrix block when decomposing A and then solving the eigenvalue problem by inverse iteration.

3. The Equations in a Matrix Block

The eigenvalue problem (1) is solved by first decomposing A into

$$A = L \cdot D \cdot L^T \tag{2}$$

followed by inverse vector iteration (see Ref. 6)

$$A\underline{x}^{k+1} = L \cdot D \cdot L^T \underline{x}^{k+1} = \underline{u}^k = B\underline{x}^k . \tag{3}$$

When performing the decomposition (2) in the original form shown in Figures 1 and 2 the non-overlapping parts of the matrix blocks fill in. The one-dimensionality of the components V and Y is destroyed. To prevent this, the decomposition of each matrix block is preceded by a rearrangement which consists of reordering the equations and renumbering the unknowns.

3.1 Rearrangement of a Matrix Block of A

The rearrangement of each matrix block is done by moving the subblocks as given in Table 1 and then renumbering the unknowns V and Y so that they alternate V_1, Y_1, V_2, Y_2, ... as shown in Figure 4. This renumbering is applied to both the rows and the columns to keep the matrix symmetric. These operations lead to the new structure shown in Figure 5. In this new block, A_1 is formed out of the original subblocks (2,2), (2,3), (3,2) and (3,3), A_2 out of (1,2) and (1,3), A_3 out of (2,4) and (3,4), A_4 out of (1,1), A_5 out of (1,4) and A_6 out of (4,4). Note that subblock (i,j) = $(j,i)^T$. The new subblock A_1 is a band matrix of band width of 15. The new subblocks A_2, A_3 and A_5 are never transformed during the calculations and it was convenient to keep them in their original forms, i.e. as off diagonal banded matrices, subblocks (1,2), (1,3), (2,4), (3,4) and (1,4) of bandwidth 7. The matrices A_4 and A_6 are considered to be full subblocks. For the i^{th} matrix block the reordered eigenvector is $\underline{x}_i$ = (Z_1,Z_2,Z_3) where $Z_1 = (V_i,Y_i)$, $Z_2 = X_i$, $Z_3 = X_{i+1}$ and the right hand side vector is $\underline{u}_i = (U_1,U_2,U_3)$. Note that Z_1, Z_2, Z_3, U_1, U_2 and U_3 are only introduced to show how the elimination in between one block is done.

3.2 The Matrix Equations of One Block

The rearranged block matrix equation is shown in Figure 5. The 3 equations can be written

$$
\begin{aligned}
A_1Z_1 + A_2Z_2 + A_3Z_3 &= U_1 \\
\ldots\ldots\ldots + A_2Z_1 + A_4Z_2 + A_5Z_3 &= U_2 \qquad (4) \\
A_3Z_1 + A_5Z_2 + A_6Z_3 + \ldots\ldots &= U_3
\end{aligned}
$$

previous block — next block

It is possible to express Z_1 as a function of Z_2 and Z_3 from the first of equations (4),

$$Z_1 = A_1^{-1}(U_1 - A_2Z_2 - A_3Z_3) \tag{5}$$

and replace it in the other two :

$$\begin{aligned} \ldots\ldots + \hat{A}_4 Z_2 + \hat{A}_5 Z_3 &= \hat{U}_2 \\ \hat{A}_5^T Z_2 + \hat{A}_6 Z_3 + \ldots\ldots &= \hat{U}_3 \end{aligned} \tag{6}$$

where

$$\begin{aligned} \hat{A}_4 &= A_4 - A_2^T \cdot A_1^{-1} \cdot A_2 & \hat{U}_2 &= U_2 - A_2^T \cdot A_1^{-1}\, U_1 \\ \hat{A}_5 &= A_5 - A_2^T \cdot A_1^{-1} \cdot A_3 & & \\ \hat{A}_6 &= A_6 - A_3^T \cdot A_1^{-1} \cdot A_3 & \hat{U}_3 &= U_3 - A_3^T \cdot A_1^{-1}\, U_1 \end{aligned} \tag{7}$$

Since the previous blocks have already been triangularized, we can continue the elimination in Eqs. (6) :

$$Z_2 = \hat{A}_4^{-1}(\hat{U}_2 - \hat{A}_5 Z_3) \tag{8}$$

and obtain the coupling part to the next block

$$\hat{\hat{A}}_6 Z_3 + \ldots = \hat{\hat{U}}_3 \tag{9}$$

where

$$\begin{aligned} \hat{\hat{A}}_6 &= \hat{A}_6 - \hat{A}_5^T \cdot \hat{A}_4^{-1} \cdot \hat{A}_5 \\ \hat{\hat{U}}_3 &= \hat{U}_3 - \hat{A}_5^T \cdot \hat{A}_4^{-1}\, \hat{U}_2 \quad . \end{aligned} \tag{10}$$

We are now ready to show how we make use of the sparseness of the matrices A_1, A_2, A_3 and A_5.

4. Implementation of the Sparse Matrix Techniques

4.1 Decomposition of the Matrix A

Before starting the inverse vector iteration, we first replace the inversions of the matrices A_1 and A_4 by their decompositions. The sparseness of A_1 is untouched when decomposing A_1 into

$$A_1 = L_1 \cdot D \cdot L_1^T . \tag{11}$$

The transformed matrix

$$\hat{A}_4 = A_4 - A_2^T \cdot A_1^{-1} \cdot A_2 = A_4 - A_2^T \cdot L_1^{-T} \cdot D_1^{-1} \cdot L_1^{-1} \cdot A_1 \tag{12}$$

is obtained through sparse matrix operations. All multiplications with the inverse of the triangular band matrices L_1 or L_1^T are replaced by solving systems of linear equations. The operations (11) and (12) are "cheap" sparse matrix equations. The transformed matrix $\hat{A}_4$ is a full matrix. It is decomposed into

$$\hat{A}_4 = \hat{L}_4 \cdot \hat{D}_4 \cdot \hat{L}_4^T . \tag{13}$$

This decomposition and the rectangular rule

$$\hat{\hat{A}}_6 = \hat{A}_6 - \hat{A}_5^T \cdot \hat{A}_4^{-1} \cdot \hat{A}_5 \tag{14}$$

are the only parts of the code where the CPU times are proportional to $N_s * N_x^3$. The matrix $\hat{\hat{A}}_6$ becomes matrix A_4 of the following block.

For the last block we have to decompose

$$\hat{\hat{A}}_6 = \hat{\hat{L}}_6 \cdot \hat{\hat{D}}_6 \cdot \hat{\hat{L}}_6^T . \tag{15}$$

4.2 The Inverse Vector Iteration

We have to solve the system of linear equations (3) repeatedly. For a given k we solve (3) in two steps :

(a) The forward reduction

The elimination to a matrix triangular form implies a forward reduction for the right hand side vector $\underline{u}^k$. For each block we have to perform (see eqs (7,10,11,12,13)) :

$$\begin{aligned}
\hat{U}_2 &= U_2 - A_2^T \cdot L_1^{-T} \cdot D_1^{-1} \cdot L_1^{-1}\, U_1 \\
\hat{U}_3 &= U_3 - A_3^T \cdot L_1^{-T} \cdot D_1^{-1} \cdot L_1^{-1}\, U_1 \qquad (16) \\
\hat{\hat{U}}_3 &= \hat{U}_3 - \hat{A}_5^T \cdot \hat{A}_4^{-1}\, \hat{U}_2 \\
&= \hat{U}_3 - (A_5^T - A_3^T \cdot L_1^{-1} \cdot D_1^{-1} \cdot L_1^{-T} \cdot A_2) \cdot \hat{L}_4^{-T} \cdot \hat{D}_4^{-1} \cdot \hat{L}_4^{-1}\, \hat{U}_2
\end{aligned}$$

b) Backsubstitution

We are now ready to solve the linear system from the bottom to the top.

For the last block we first solve for eqs. (9,10,15)

$$Z_3 = \hat{\hat{L}}_6^{-T} \cdot \hat{\hat{D}}_6^{-1} \cdot \hat{\hat{L}}_6^{-1}\, \hat{\hat{U}}_3 \quad , \qquad (17)$$

For all blocks we solve from the bottom to the top (see eqs. (5-8,11-13, 16,17)) :

$$
\begin{aligned}
Z_2 &= \hat{L}_4^{-T} \cdot \hat{D}_4^{-1} \cdot \hat{L}_4^{-1} \, (\hat{U}_2 - (A_5 - A_2^T \cdot L_1^{-T} \cdot D_1^{-1} \cdot L_1^{-1} \cdot A_3) \, Z_3) \\
Z_1 &= L_1^{-T} \cdot D_1^{-1} \cdot L^{-1} (U_1 - A_2 Z_2 - A_3 Z_3) \qquad (18) \\
Z_3 &= Z_2 \quad .
\end{aligned}
$$

At the end of the iteration we reorder the final eigenvector back to its original form.

5. The Gain due to Sparseness Techniques

5.1 CPU time

The CPU time in seconds for a CYBER 170-720 of the published version T_P can be expressed in terms of the number of intervals in radial direction (N_s = number of matrix blocks), in poloïdal direction (N_χ, $8N_\chi + 8$ = dimension of one original matrix block, $2N_\chi + 2$ = region of overlap) and in terms of the number of iterations N_{it}

$$
T_P \simeq N_s \, (N_\chi + 1)^2 * \left[7.5 * 10^{-4} (N_\chi + 1) + 2.7 * 10^{-3} \, N_{it} \right] \quad \text{sec} \qquad (19)
$$

For the new version which includes the described sparse matrix operations, the CPU time T_s can be approximated by

$$
\begin{aligned}
T_S \simeq N_s \, (N_\chi + 1) \Big[& 2*10^{-4} (N_\chi + 1)^2 + 6.4*10^{-3} (N_\chi + 1) + 3.1*10^{-2} + \\
& + N_{it} (10^{-4} (N_\chi + 1) + 0.02) \Big] \quad \text{sec} \qquad (20)
\end{aligned}
$$

Comparisons between CPU time of the published and the new version are made in Table 2 for the eigenvalue solver. Note that the whole ERATO code includes also the calculation of the mapping, the vacuum, the matrix elements, and the diagnostics. The CPU time of these parts strongly depends on the cases one runs. However for large meshes the total CPU time is dominated by the eigenvalue calculation. When we consider the asymptotic gain coming from the decomposition for $N_\chi \longrightarrow \infty$, TP/TS = 3.75. This factor can be explained in the following way :

The number of rectangular rules M_p we have to perform for a whole filled in block in the published version is ($N = 2N_\chi + 2$)

$$M_p = 10N^3 + N\cdot(N^2-1)/2 \ . \qquad (21)$$

Here we did not take into account that the upper triangles of the blocks (1,2), (1,3) and 1,4) (see Figure 1) are empty. The published version takes advantage of these zeros. A reduction of the CPU time of about a factor of 2 is observed. The number of rectangular rules M_s for the new version is given by

$$M_s = N^3 + N\cdot(N^2-1)/2 \ . \qquad (22)$$

The leading terms for $N \longrightarrow \infty$ are $M_p = 63/6$ and $M_s = 7/6$. If we could not take advantage of appearing zeros in the published version, a theoretical gain of 9 could be obtained for $N \longrightarrow \infty$. In reality this gain is 3.75 as we have seen above. We have seen that this gain in CPU time can only be obtained on a scalar machine. On a vector machine such as a CRAY1 no gain in CPU time in respect to the HYMNIABLOCK /3/ subroutines has been observed. This is due to the short band widths of the banded matrices which unables us to make full use of the vectorization.

5.2 Memory storage

The memory space required by in the published (MP) and the new (MS) version is given by

$$\begin{aligned} MP &\cong 12700 + 2(2N_\chi + 2)(3N_s + 16N_\chi + 23) \\ MS &\cong 16800 + (2N_\chi + 2)(3N_s + 4N_\chi + 69) \end{aligned} \tag{23}$$

Table 3 shows the memory space used in the eigenvalue solver for different mesh sizes. For $N_s = N_\chi - 1 \longrightarrow \infty$, MP/MS = 5.4.

5.3 Disk storage

Disk storage in the published (DP) and in the new (DS) version are given by

$$\begin{aligned} DP &\cong 174\ 000 + 48N_sN_\chi\ (2N_\chi + 5) \\ DS &\cong 140\ 000 + N_sN_\chi\ (2N_\chi + 364) \end{aligned} \tag{24}$$

The constants 174 000 and 140 000 contain all the CDC-UPDATE files, i.e. SOURCE, COMPILE, LGO for all the 5 main programs of ERATO. Table 4 shows the disk storage for both versions for different mesh sizes.

For $N_s = N_\chi - 1 \longrightarrow \infty$, DP/DS = 48.

5.4 Input - output operations

The amount of input - output operations is mainly defined by the number of iterations where the decomposed matrix blocks have to be read $2*N_{it}$ times. For the published (IOP) and the new (IOS) versions the number of words to be read during decomposition, iteration and to perform the Rayleigh quotient is given by

$$IOP \cong 32\ N_s\ (N_x + 1)^2\ \left[5 + 4N_{it}\right] \tag{25}$$

$$IOS \cong 4\ N_s\ (N_x + 1)\ \left[135 + N_{it}(N_x + 108)\right]$$

Table 5 shows the number of words which have to be read in from disk in the eigenvalue solver. Asymptotically, for $N_s = N_x - 1 \longrightarrow \infty$, IOP/IOS $>$ 32. It depends on the number of iterations.

6. Conclusion

In the variational form of the ideal linear MHD operator it is possible to find a coordinate system as well as vector components such that there are no radial derivatives on two of the three components. By choosing a piecewise constant basis in the r-direction for these components the resulting matrix eigenvalue problem has a particular form. Half of each matrix block does not overlap with the previous or with the following block. Sparse matrix techniques can be used to take advantage of this blockwise "one-dimensionality" of the two components. As techniques we apply matrix reordering, renumbering of the unknowns and solving linear systems instead of inverting matrices. The gains in CPU time and memory storage is of order of 4. The gain in disk storage and input - output operations is of order of 10. For a vector machine no gain in CPU time has been observed. However the gain in memory space, disk storage and IO operations remains and reduces drastically the turn-around time on such a machine.

Acknowledgements

This work was supported by the Fusion Energy Division of the Oak Ridge National Laboratory, the Ecole Polytechnique Fédérale of Lausanne, the Swiss National Science Foundation and Euratom.

References

/1/ R. Gruber, F. Troyon, D. Berger, L.C. Bernard, S. Rousset, R. Schreiber, W. Kerner, W. Schneider, K.V. Roberts, Computer Phys. Commun. 21, 323 (1981)

/2/ Y. Tanaka, T. Matsuura, T. Takeda, M. Azumi, S. Tokuda, T. Tsunematsu, G. Kurita, T. Takizuka, JAERI-M 9040 (1980)

/3/ R. Gruber, Computer Phys. Commun. 20, 421 (1980)

/4/ R. Gruber, S. Rousset, F. Troyon, W. Kerner, L.C. Bernard, Computer Phys. Commun. 22, (1981)

/5/ I.B. Bernstein, E.A. Frieman, M.O. Kruskal, R.M. Kulsrud, Proc. Roy. Soc. (London), Ser. A244 (1968)

/6/ J.H. Wilkinson, The Algebraic Eigenvalue Problem, (Clarendon Press, Oxford 1965)

Table 1 : Rearrange matrix block of A	
Subblock	becomes
(1,1)	(3,3)
(1,2)	(3,1)
(1,3)	(3,2)
(1,4)	(3,4)
(2,2)	(1,1)
(2,3)	(1,2)
(2,4)	(1,4)
(3,3)	(2,2)
(3,4)	(2,4)

Table 2 : CPU-times in seconds (N_{it} = 4) on CYBER 170-720

N_s = N_x -1	Published version (P)	New version (S)	P/S
10	29	27	1.1
20	260	160	1.6
30	1 100	500	2.1
40	3 000	1 300	2.3
50	6 700	2 600	2.6
60	13 000	4 800	2.7
70	24 000	8 300	2.9
80	39 000	13 000	3.0
90	61 000	20 000	3.1
100	91 000	29 000	3.1

Table 3 : Memory storage (60 bits words)

$N_s = N_x - 1$	Published version (P)	new version (S)	P/S
10	23 700	20 300	1.2
20	50 000	26 200	1.9
30	90 700	35 000	2.6
40	147 000	46 500	3.2
50	218 500	60 800	3.6
60	305 100	78 000	3.9
70	407 000	97 900	4.2
80	524 100	120 700	4.3
90	656 400	146 200	4.5
100	803 900	174 500	4.6

Table 4 : Disk storage in Kwords

$N_s = N_x - 1$	Published version (P)	New version (S)	P/S
10	317 K	183 K	1.7
20	1 222 K	311 K	3.9
30	3 165 K	537 K	5.9
40	7 023 K	872 K	8.1
50	13 271 K	1 329 K	10.0
60	22 486 K	1 919 K	11.7
70	35 243 K	2 655 K	13.3
80	52 118 K	3 549 K	14.7
90	73 688 K	4 612 K	16.0
100	100 371 K	5 914 K	17.0

Table 5 : input - output operations ($N_{it} = 4$) in Kwords

$N_s = N_x - 1$	Published version (P)	New version (S)	P/S
10	968 K	297 K	3.3
20	6 505 K	1 146 K	5.7
30	20 644 K	2 654 K	7.8
40	47 417 K	4 913 K	9.7
50	90 855 K	8 019 K	11.3
60	154 991 K	12 068 K	12.8
70	243 856 K	17 157 K	14.2
80	361 483 K	23 380 K	15.5
90	511 903 K	30 835 K	16.6
100	599 149 K	39 617 K	17.6

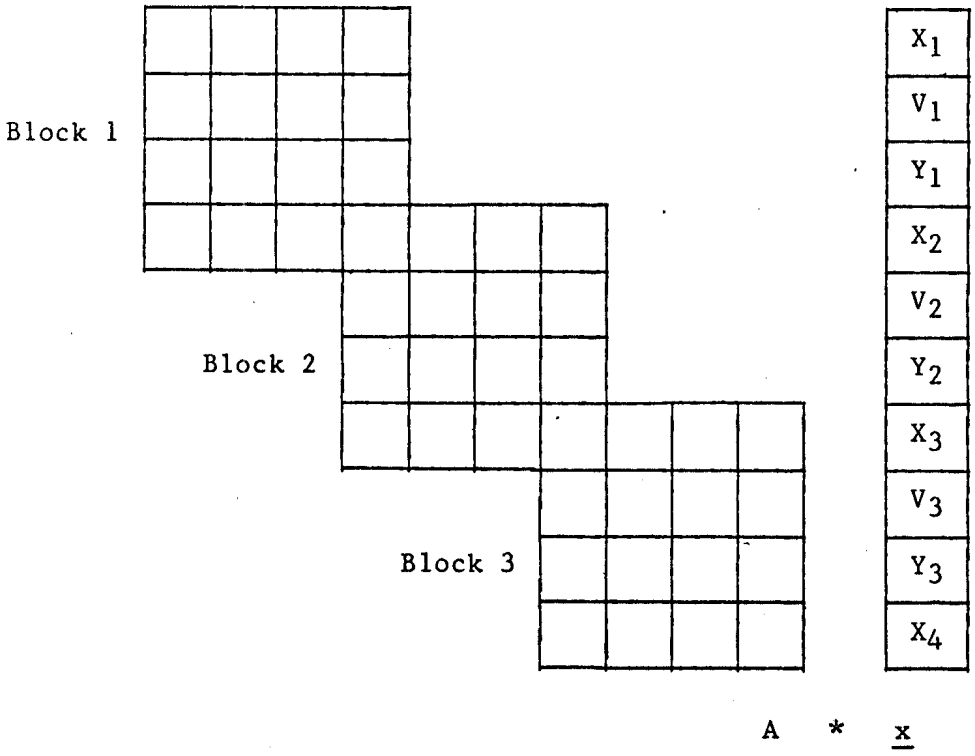

FIGURE 1

The left hand side part $\underline{Ax}$ of the eigenvalue problem $\underline{Ax} = \omega^2 \underline{Bx}$. $N_s = 3$.

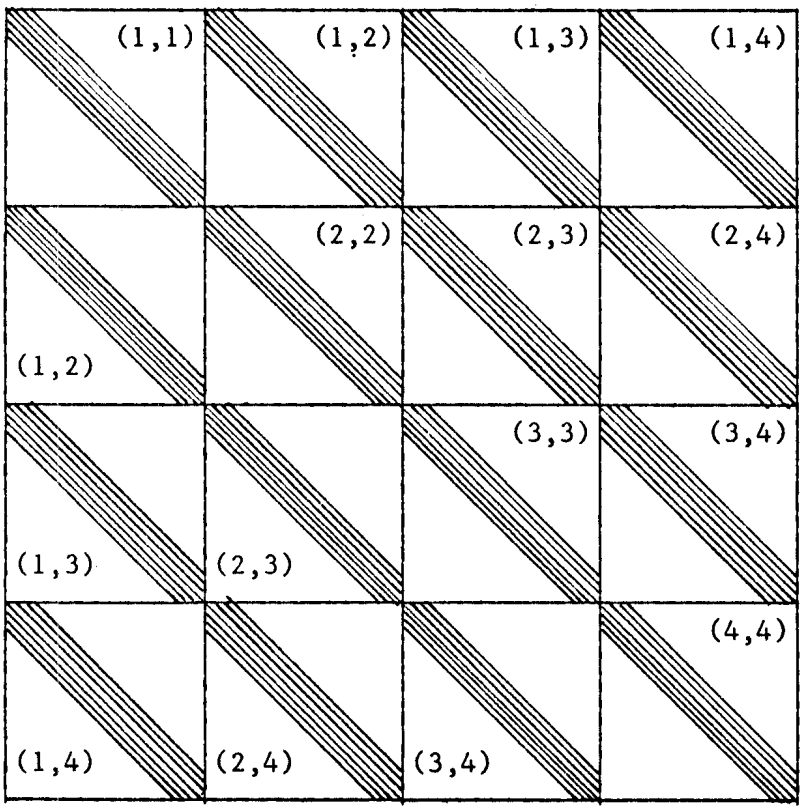

FIGURE 2.

Structure of one matrix block.

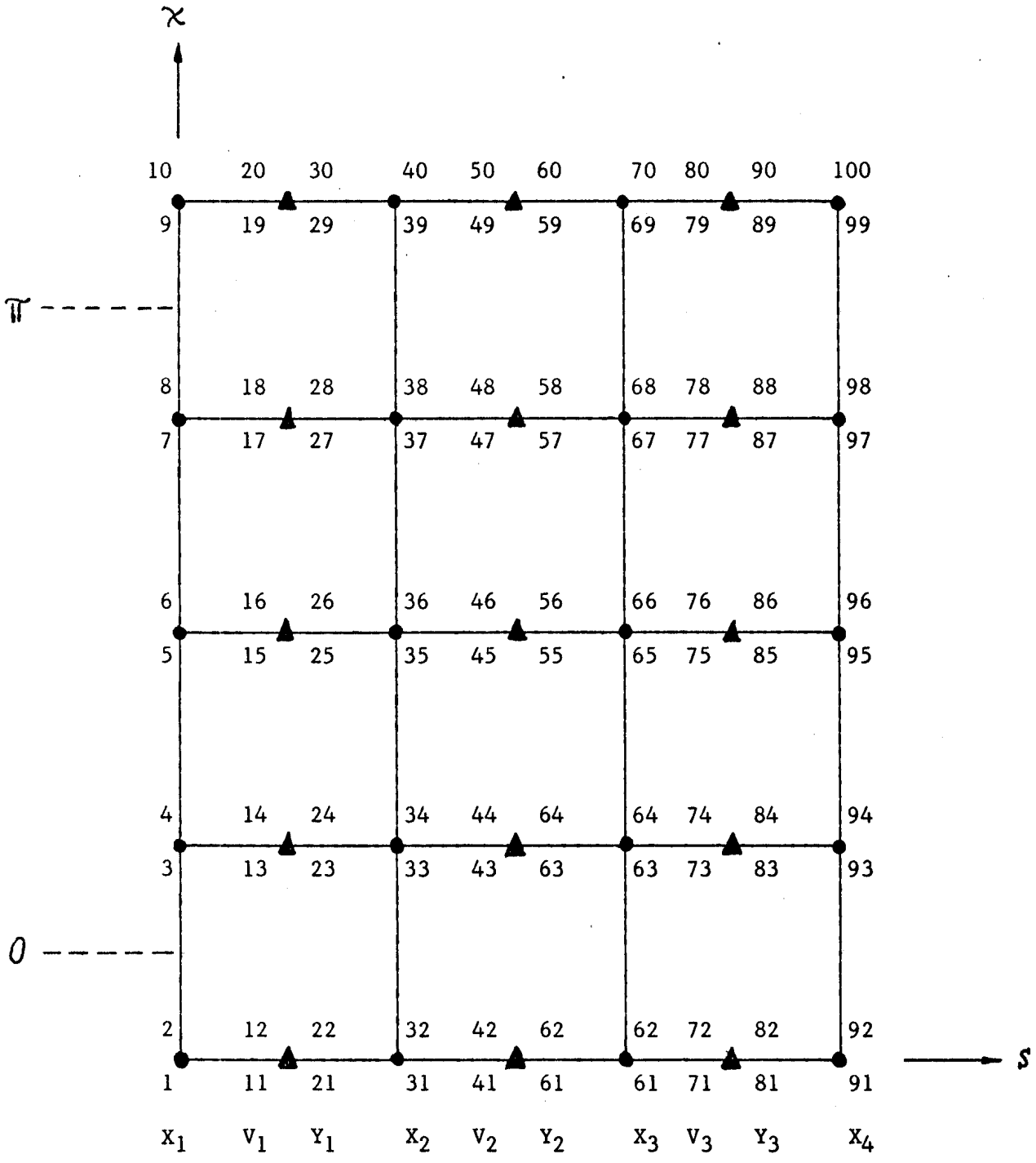

FIGURE 3.

Initial numbering. Two unknowns per nodal point for real and imaginary part. $N_s=3$, $N_\chi=4$.

● Nodal positions of X.

▲ Nodal positions of V and Y.

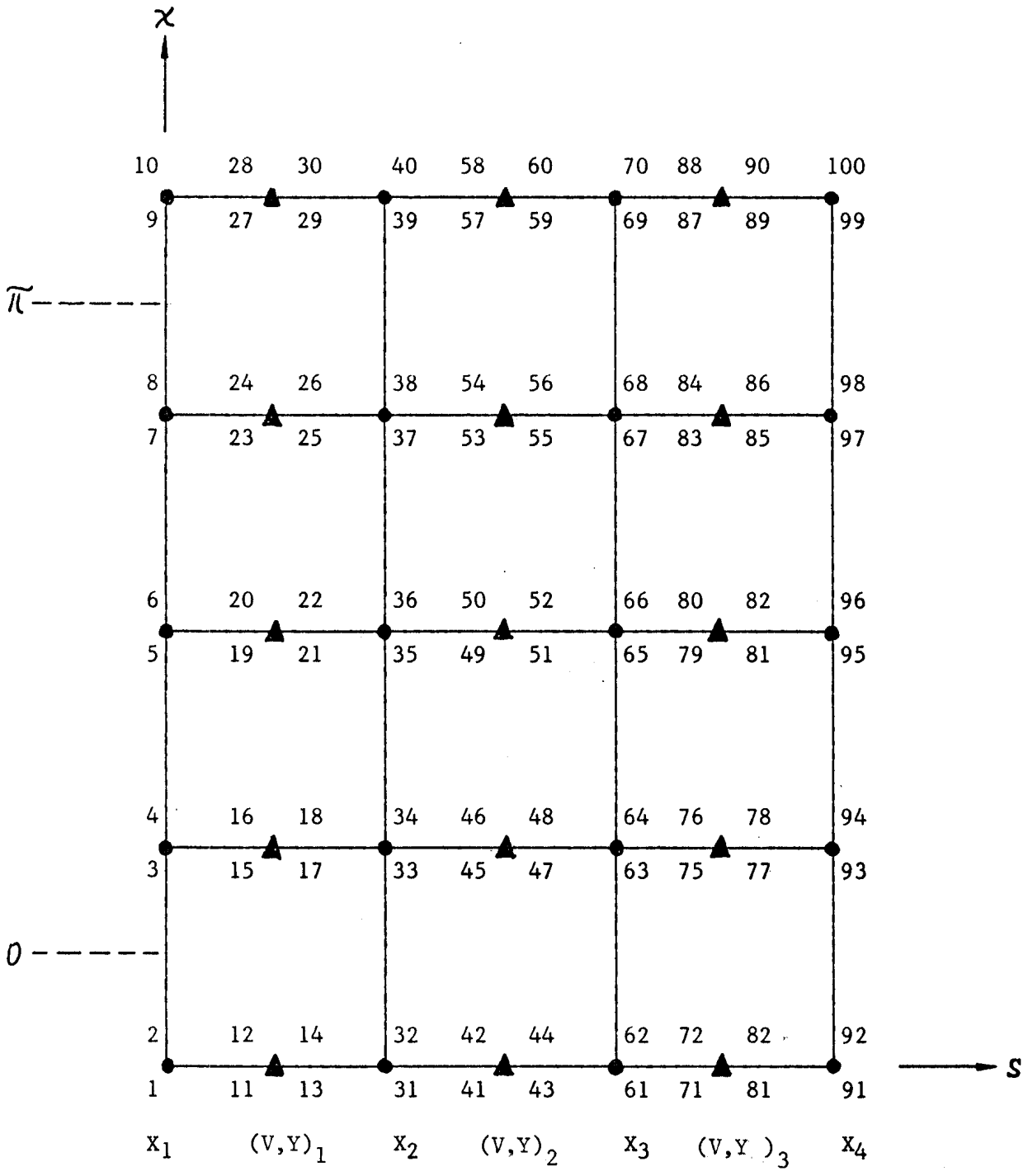

FIGURE 4.

Renumbering. N_s=3. N_x =4.

● Nodal positions of X.

▲ Nodal positions of (V,Y).

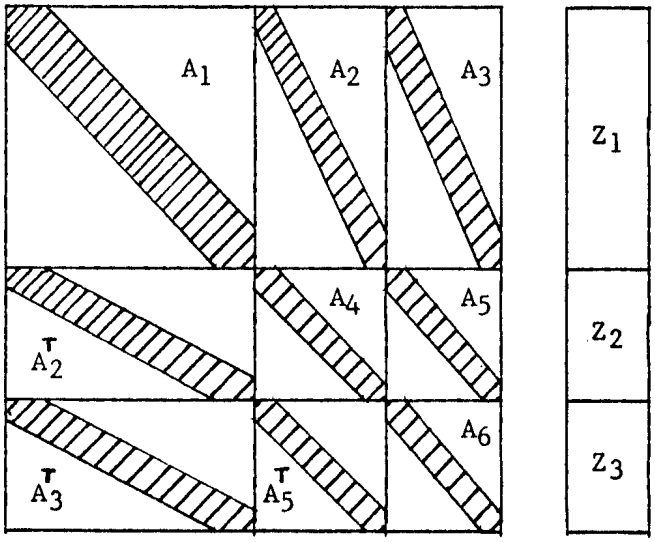

FIGURE 5.

Rearranged structure of the i^{th} matrix block of A multiplied with the reordered part of the eigenvector which multiplies this matrix. $Z_1 = (V_i, Y_i)$, $Z_2 = X_i$, $Z_3 = X_{i+1}$.