DR. 1056

ORNL/CSD/TM-117

202
4/16/60
UCC-ND

NUCLEAR
DIVISION

UNION
CARBIDE

MASTER

**Implementing Sparse Matrix
Techniques in the ERATO Code**

D. S. Scott

OPERATED BY
UNION CARBIDE CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

ORNL/CSD/TM-117
Distribution Category UC-32

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any of its employees makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

IMPLEMENTING SPARSE MATRIX
TECHNIQUES IN THE ERA/O CODE

D. S. Scott

Sponsor: D. A. Gardiner
Originator: D. S. Scott

Date Published - April 1980

NOTICE This document contains information of a preliminary nature. It is subject to revision or correction and therefore does not represent a final report.

COMPUTER SCIENCES DIVISION
at
Oak Ridge National Laboratory
Post Office Box Y
Oak Ridge, Tennessee 37830

Union Carbide Corporation, Nuclear Division
operating the
Oak Ridge Gaseous Diffusion Plant . Paducah Gaseous Diffusion Plant
Oak Ridge Y-12 Plant . Oak Ridge National Laboratory
under Contract No. W-7405-eng-26
for the
Department of Energy

UNCLASSIFIED

224

CONTENTS

LIST OF ILLUSTRATIONS.	v
1. INTRODUCTION	1
2. FACTORING $A - \omega_0 B$	2
2.1 Reordering Scheme	3
3. SOLVING EQUATIONS WITH THE FACTORED FORM	5
3.1 Forward Reduction	5
3.2 Back Substitution	6
4. TWO IMPROVEMENTS	6
5. ARRAY STORAGE AND TRANSFORMATIONS.	7
6. PACKING ARRAYS	10
7. SUBROUTINE STRUCTURE	11
7.1 High Level Subroutines.	11
7.2 Specific Task Modules in the Factorization Phase.	12
7.3 Specific Task Modules in the Solution Phase	12
7.4 Packing Subroutines	13

ILLUSTRATIONS

Figure

1	Overlapping Block Diagonal Structure.	15
2	Subblocks of a Block.	15
3	Subblock Structure.	16
4	The Nonzero Structure of a Typical Block of the Factored Matrix.	16
5.	Original Matrix	18
6.	Reordered Matrix.	19
7.	Storage of ALBAND	20
8.	Storage in OFDIAG	21
9.	Storage in DTR.	22

1. INTRODUCTION

The ERATO code [1] computes the stability of an equilibrium with respect to the linear MHD equation. The ERATO code is divided into 5 programs (called ERAT01 through ERAT05). In this report we document some minor changes made in ERAT03 and a major reorganization of ERAT04. The changes were made to drastically reduce the amount of secondary storage needed by the codes and at the same time improve the efficiency of ERAT04. The ultimate goal is to allow the successful completion of runs with finer meshes than are currently possible.

ERAT04 takes given matrices A and B and a shift parameter ω_0 and computes the Choleski factorization ($U^T D U$) of the matrix $A - \omega_0 B$. It then uses the factorization to implement inverse iteration to find the eigenvalue of the generalized eigenvalue problem $A - \lambda B$ closest to ω_0 and its corresponding eigenvector.

The matrices A and B are highly structured and quite sparse. Both are symmetric and B is positive definite. Both matrices have an overlapping block diagonal structure with NPSI blocks. Each block has dimension $8 \cdot NCHI + 8$ and consists of sixteen square subblocks of dimension $2 \cdot NCHI + 2$. Each subblock has a block tridiagonal structure with 2×2 blocks. In the A matrix these 2×2 blocks are usually full while in the B matrix the 2×2 blocks are always diagonal so that the B matrix has about half the number of nonzeros as the A matrix.

The diagonal blocks overlap by one subblock so that the (1,1) subblock of one block is the (4,4) subblock of the previous block while the (4,4) subblock of the block is the (1,1) subblock of the next block.

In each block, some of the elements of the matrices described as non-zero above are actually zero. In particular the first three off diagonal subblocks in the first block are identically zero. However no advantage is taken of these special zeros in the coding and they will not be mentioned again.

On the other hand, in some cases the last subblock of the last block will have additional nonzeros (vacuum terms) and will be a full triangle (because of the symmetry of the matrices it is only necessary to store the upper triangular part). Rather than try to distinguish cases, the code assumes that this last subblock is always a full triangle.

The structure of the matrices is shown in Figures 1, 2, and 3.

2. FACTORING $A - \omega_0 B$

The original ERATO4 factors the matrix $A - \omega_0 B$ in its natural order. This causes the blocks to completely fill in (become nonzeros) except for half of three subblocks (see Figure 4). In this section we describe a method of factorization which greatly reduces the number of nonzeros which must be stored. This method uses two well known sparse matrix techniques: reordering the variables and equations to reduce fill in and storing matrices in factored form.

To maintain the block structure of the matrix it is necessary that the (4,4) subblock not be changed by the reordering (since that subblock is coupled to the next block) but it is permissible to move the (1,1) subblock since the coupling terms in the previous block have already been eliminated. This elimination in the earlier block causes the (1,1) subblock to be full. (Except in the first block but again no special provision is made for this fact - the code assumes that the (1,1) subblock is full even in the first block.

To maintain symmetry it is necessary to apply the same permutation to both the rows and the columns of the matrix. Therefore to describe the reordering procedure it is sufficient to describe the new order in which the diagonal elements of the block appear.

REORDERING SCHEME

1. Permute rows and columns so that the (1,1) subblock appears as the (3,3) subblock.
2. Interleave the (2,2) and (3,3) subblocks to appear in the (1,1) and (2,2) subblocks. That is, take the first diagonal element of the (2,2) subblock, then the first diagonal element of the (3,3) subblock, then the second diagonal element of the (2,2) subblock, and so on. This interleaving causes the (2,2), (2,3), and (3,3) subblocks to combine into a long band matrix.

The result of this reordering is shown in Figure 5 and Figure 6.

The new block has a 3 x 3 subblock structure. The (1,1) subblock is twice as big as before and has a half band width of eight. The (1,2) and (1,3) subblocks are rectangular and each is composed of the interleaving of two old off-diagonal subblocks. It should be noted that the old off-diagonal blocks that make up the new (1,2) subblock are transposed before they are assembled because the old (1,1) subblock has gone from being above the old (2,2) and (3,3) subblocks to being below them. The (2,2) subblock is the old (1,1) subblock, the (2,3) subblock is the old (1,4) subblock and the (3,3) subblock is the old (4,4) subblock.

After factoring the reordered block the following pieces are stored.

1. The factored (1,1) subblock.

2. The unfactored offdiagonal blocks.
3. The factored (2,2) subblock.

See Figures 7,8, and 9 for a further description of the stored information. the (3,3) subblock is altered by the factorization (filled in) but is not itself factored or stored. It is passed on to the next block instead.

For convenience, consider the reordered block to consist of 4 subblocks by combining the (1,2) and (1,3) subblocks and combining the (2,2), (2,3), and (3,3) subblocks.

Label the subblocks as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix}.$$

Then the factoring scheme is as follows.

1. Factor A_{11} (a band matrix).
2. Compute $\hat{A}_{22} = A_{22} - A_{12}^T A_{11}^{-1} A_{12}$.

(the Schur complement of A_{22} - sometimes called the reduced or active matrix)

3. Factor the first half of $\hat{A}_{22}$.
4. Store the appropriate information.
5. Pass the (3,3) subblock to the next block.

Of course it is necessary to factor and store the (3,3) subblock when factoring the last block.

3. SOLVING EQUATIONS WITH THE FACTORED FORM

In Section 2 we described the information which is stored after factoring each block. In this section we show that the stored information is sufficient to solve linear equations by listing the sequence of operations needed to do so. For notation the factored form of the (1,1) subblock is $U_1^* D_1 U_1$, the factored form of the (2,2) subblock is $U_2^* D_2 U_2$. Finally, the three off diagonal blocks of the unfactored matrix are called A_{12} , A_{13} , and A_{23} respectively. (A_{12} and A_{13} are rectangular while A_{23} is square.) The vector x is partitioned conformally into x_1 , x_2 , and x_3 . x_1 is twice as long as x_2 or x_3 .

The solution procedure involves a forward reduction (including division by the diagonal matrix) and a backsubstitution. In the following w is a scratch vector used to hold an intermediate vector quantity.

FORWARD REDUCTION

1. Reorder active part of x .
2. $x_1 \leftarrow D_1^{-1} U_1^{-T} x_1$
3. $w \leftarrow U_1^{-1} x_1$
4. $x_2 \leftarrow x_2 - A_{12}^T w$
5. $x_3 \leftarrow x_3 - A_{13}^T w$
6. $x_2 \leftarrow D_2^{-1} U_2^{-T} x_2$
7. $x_3 \leftarrow x_3 - A_{23}^T - A_{13}^T U_1^{-1} D_1^{-1} U_1^{-T} A_{12}) U_2^{-1} x_2$

After the forward reduction of each block it is necessary to do the forward reduction of the bottom subblock

$$1. \quad x_3 \leftarrow D_3^{-1} U_3^{-T} x_3$$

Then the back substitution for the bottom subblock is performed.

$$1. \quad x_3 \leftarrow U_3^{-1} x_3$$

Then each block is visited in reverse order for the back substitution.

BACK SUBSTITUTION

$$1. \quad x_2 \leftarrow x_2 + D_2^{-1} U_2^{-T} (-A_{23} + A_{12}^T U_1^{-1} D_1^{-1} U_1^{-T} A_{13}) x_3$$

$$2. \quad x_2 \leftarrow U_2^{-1} x_2$$

$$3. \quad x_1 \leftarrow x_1 - D_1^{-1} U_{11}^{-T} (A_{12} x_2 + A_{13} x_3)$$

$$4. \quad x_1 \leftarrow U_{11}^{-1} x_1$$

5. Return to original order.

The factored form of the operators looks rather messy but every one of the matrices appearing in the formulas is sparse except U_2 so that the matrix vector products can be computed quickly.

4. TWO IMPROVEMENTS

Two improvements were made in ERAT04. In subroutine BFIX, the array B is read in by blocks and written out by diagonals. This allows the multiplication of a vector by B to vectorize easily.

In the current ERAT04 the starting vector for the vector iteration is taken to be the vector of all ones. The new version chooses a starting

vector based on the factorization of $A - \omega_0 B = U^T D U$. The location of the algebraically smallest element of D is saved from the factorization. If w is a vector of all zeros except for a 1 in the corresponding location, then the starting vector is taken to be $= U^{-1} w$. This starting vector is especially good if the element of D is negative (as will happen whenever the shift ω_0 is larger than the eigenvalue. In this case $x^* A x / x^* B x$ will already be smaller than ω_0 .

5. ARRAY STORAGE AND TRANSFORMATIONS

ERAT04 has a large storage array (of dimension at least as large as a triangular block) in common. In coding the new algorithm parts of this array were used to store various pieces of the active block in the factorization and solution process. The following names are consistently used throughout the code. See Figures 7, 8, and 9 for further description.

NS $2 * NCHI + 2$, the dimension of a subblock

NS2 $2 * NS$, the dimension of the (1,1) subblock in the reordered matrix.

ALBAND(8,NS2) used to store the transpose of the (1,1) subblock of the reordered matrix. The first row is the main diagonal. The columns correspond to rows. This matrix is factored in place.

OFDIAG(NS,7,5) This matrix holds five off diagonal subblocks of the original matrix. The 7 columns of each plane are the seven diagonals of the subblock. The stored rows are the rows of the subblock. (OFDIAG(*,*,1) stores the transpose of the original (1,2) subblock.

OFDIAG (*, *, 2) stores the original (2,4) subblock.
 OFDIAG (*, *, 3) stores the transpose of the
 original (1,3) subblock, OFDIAG (*, *, 4) stores the
 original (3,4) subblock. OFDIAG (*, *, 5) stores
 the original (1,4) subblock. Note that the off-
 diagonal blocks are not interleaved for storage pur-
 poses.

DSQ(NS2, NS2) is used for intermediate quantities in computing
 $\hat{A}_{22}$ in the factorization process. It then holds
 $\hat{A}_{22}$ for factorization.

DTR(i, S2 (NS2+1)/2) is used to hold A_{22} . $\hat{A}_{22}$ is assembled in
 it. It is then transferred to OSQ for factorization.
 CA(NS(NS+1)/2) is used to hold C_2 and U_2 for storage purposes.
 It then holds the (3,3) subblock (unfactored) for
 transfer to the next block.

Only ALBAND, OFDIAG, and CA are needed for the solution phase.

We now give the formulas for the transfer of elements from the
 original block to the arrays ALBAND, OFDIAG, and DTR. Notation:
 $NS3 = NS2 + NS$ and $NS4 = NS3 + NS$. We label the subblocks of the upper
 triangle of a block so that subblocks 1, 5, 8, and 10 are on the diagonal.

ALBAND

$$A_{IJ} \rightarrow \text{ALBAND} (2*(J-I)+1, 2*(I-NS)-1)$$

$$\text{for } NS + 1 \leq I \leq NS2$$

$$\leq J \leq NS2$$

(Subblock 5)

$$A_{IJ} \rightarrow \text{ALBAND} (2 * (J-I) + 1, 2 * (I-NS2))$$

$$\text{for } NS2 + 1 \leq I \leq NS2$$

$$I \leq J \leq NS3$$

(Subblock 8)

$$A_{IJ} \rightarrow \text{ALBAND} (2 * (J-I-NS) + 2, 2 * (I-NS) - 1)$$

$$\text{for } NS + 1 \leq I \leq NS2$$

$$I \leq J \leq NS2$$
(upper triangle
of Subblock 6)
$$A_{IJ} \rightarrow \text{ALBAND} (2 * (I-J+NS), 2 * (J-NS2))$$

$$\text{for } NS + 1 \leq I \leq NS2$$

$$NS + 1 \leq J < I$$
(strictly lower
triangle of subblock
6)OFDIAG

$$A_{IJ} \rightarrow \text{OFDIAG} (J-NS, 4-J+I+NS, 1)$$

$$\text{for } 1 \leq I \leq NS$$

$$NS + 1 \leq J \leq NS2$$

(subblock 2)

$$A_{IJ} \rightarrow \text{OFDIAG} (J-NS2, 4-J+I+NS2, 3)$$

$$\text{for } 1 \leq I \leq NS$$

$$NS2 + 1 \leq J \leq NS3$$

(subblock 3)

$$A_{IJ} \rightarrow \text{OFDIAG} (I, 4+J-I-NS3, 5)$$

$$\text{for } 1 \leq I \leq NS$$

$$NS3 + 1 \leq J \leq NS4$$

(subblock 4)

$$A_{IJ} \rightarrow \text{OFDIAG} (I-NS, 4+J-I-NS2, 2)$$

$$\text{for } NS + 1 \leq I \leq NS2$$

$$NS3 + 1 \leq J \leq NS4$$

(subblock 7)

$A_{IJ} \rightarrow \text{OFDIAG}(I - \text{NS2}, 4 + J - I - \text{NS}, 4)$
 for $\text{NS2} + 1 \leq I \leq \text{NS3}$
 $\text{NS3} + 1 \leq J \leq \text{NS4}$ (subblock 9)

DTR

The last element of the $(I-1)$ th row of DTR is DTR $[R(I)]$ where

$$R(I) = (I-1) * \text{NS2} - (I-1) * (I-2)/2.$$

Hence:

$A_{IJ} \rightarrow \text{DTR}(R(I) + J - I + 1)$
 for $1 \leq I \leq \text{NS}$
 $I \leq J \leq \text{NS}$ if first block (subblock 1)

$A_{IJ} \rightarrow \text{DTR}(R(I) + J - I + 1)$
 for $\text{NS3} + 1 \leq I \leq \text{NS4}$
 $I \leq J \leq \text{NS4}$ (subblock 10).

For every block except the first, subblock 1 is supplanted by the subblock brought forward from the previous block (stored in CA).

For every block, subblock 4 is copied into DTR row by row as follows

For $J = 1, \text{NS}$
 $\text{OFDIAG}(I, J, 5) \rightarrow \text{DTR}(R(I) + \text{NS} - 4 + J)$ for $I = 1, 7.$

6. PACKING ARRAYS

In ERAT03 the matrices A and B are created block by block and written out as full triangles. To reduce the amount of disk writing, it was decided to write these matrices in packed form by removing the zeros. The subroutine WRPACK is used to compress the full triangle

while creating an integer array to point to the original locations. A subroutine WR is then used to write out NZ, the number of nonzeros, followed by the packed array and the integer pointer array.

Subroutines RDPACK and RD reverse this process to recover the full triangular block.

In ERATO4, RDPACK is not used. The subroutine RD is used to obtain the packed form directly. This packed form is unpacked in whatever manner is appropriate for the given application. For example, in subroutine BSHFT the packed arrays are read in, subtracted, and rewritten without ever fully unpacking them.

7. SUBROUTINE STRUCTURE

There are five classes of subroutines.

1. High level subroutines

BSHFT: This computes $A - \omega_0 B$ and stores it in packed form.

FACMAT: This organizes the factorization of the matrix.

BFIX: This rewrites B by diagonals to improve vectorization of the matrix-vector product calculation.

XFIX: This computes the special starting vector as described in Section 4.

BMULT: This computes Bx .

FREDUC: This performs the forward reduction phase of the solution procedure (including the forward reduction of the last sub-block) and the division by the diagonal.

BACSUB: This performs the back substitution.

2. Specific task modules in the factorization phase.

RDMAT: This reads a packed block of the matrix and distributes the elements to the various arrays (as described in Section 5).

FACBND: This factors ALBAND.

FIXOSQ: This sets OSQ to A_{12}^T (as defined in Section 2)

ALBOSQ: This overwrites the A_{12}^T in OSQ by $(A_{11}^{-1} A_{12})^T = A_{12}^T U_1^{-1} D_1^{-1} U_1^{-T}$

OFDOSQ: This subroutine subtracts from DTR (which contains the upper triangle of A_{22}) the upper triangle of $(A_{12}^T A_{11}^{-1}) A_{12}$ to create the upper triangle of $\hat{A}_{22}$. It then copies DTR into the lower triangle of OSQ.

FACOSQ: This factors either the first half or all of OSQ.

CACOPY: This is used to extract diagonal subblocks (triangles) from OSQ.

PUTMAT: This stores the appropriate pieces of the factored block.

3. Specific task modules in the solution phase.

GETMAT: This reads in the pieces of the factored block.

LBDDSL: This solves a lower triangular band system (ALBAND) including division by the diagonal.

UBDSOL: This solves an upper triangular band system (ALBAND).

LTRDSL: This solves a lower triangular system (CA) including division by the diagonal.

UTRSOL: This solves an upper triangular system (CA).

OFDMILT: This multiplies a vector by A_{12} or A_{13} (as defined in Section 3).

ODTMLT: This multiplies a vector by A_{12}^T or A_{13}^T .

OFD2MT: This multiplies a vector by A_{23} .

OD2TMT: This multiplies a vector by A_{23}^T .

4. Packing subroutines.

RDPACK: This unpacks a block of a matrix into a triangle.

RD: This reads the packed block.

WPACK: This packs up a triangular block of a matrix.

WR: This writes the packed block.

5. Basic Linear Algebra Subroutines

SAXPY: This adds a scalar A times a vector X to a vector Y.

SCOPY: This copies a vector X to onto vector Y.

SDOT: This computes the inner product of two vectors.

SSCAL: This multiplies a vector X by a scalar A.

SXYPZ: This compute $Z = Z + X * Y$ elementwise.

VZERO: This zeros vector X.

Note that in all the above routines except VZERO the vectors need not be stored sequentially in memory. They need only have a constant increment.

Figure 1. Overlapping block diagonal structure.

Figure 2. Subblocks of a block.

Figure 3. Subblock structure.

Figure 4. The nonzero structure of a typical block of the factored matrix.

Key to Figures 5, 6, 7, 8, and 9

We label the subblocks of the upper triangle of a block by row so that subblocks 1, 5, 8, and 10 are the diagonal subblocks. Each nonzero element of a subblock is given a unique upper or lower case letter. Different type faces are used to distinguish subblocks which are mixed together by the reordering process

light italic (LICHT ITALIC) Subblocks 2, 4, 6, 9

letter gothic (LETTER GOTHIC) Subblocks 1, 7, 8

prestige elite (PRESTIGE ELITE) Subblocks 3, 5, 10

FIGURE 5 shows the original matrix

FIGURE 6 shows the reordered matrix

FIGURE 7 shows the storage of elements in ALBAND.

FIGURE 8 shows the storage of elements in OFDIAG.

FIGURE 9 shows the storage of elements in DTR.

A B C D E F G H	A B C D	A B C D	A B C D
J K L M N P Q	E F G H	E F G H	E F G H
R S T U V W	J K L M N P	J K L M N P	J K L M N P
X Y Z a b	Q R S T U V	Q R S T U V	Q R S T U V
c d e f	W X Y Z a b	W X Y Z a b	W X Y Z a b
g h i	c d e f g h	c d e f g h	c d e f g h
1 j k	2 i j k m	3 i j k m	4 i j k m
m	n p q r	n p q r	n p q r
A B C D	A B C D	A B C D	
E F G	E F G H	E F G H	
H J K L	J K L M N P	J K L M N P	
M N P	Q R S T U V	Q R S T U V	
Q R S T	W X Y Z a b	W X Y Z a b	
U V W	c d e f g h	c d e f g h	
5 X Y	6 i j k m	7 i j k m	
Z	n p q r	n p q r	
	A B C D	A B C D	
	E F G	E F G H	
	H J K L	J K L M N P	
	M N P	Q R S T U V	
	Q R S T	W X Y Z a b	
	U V W	c d e f g h	
	8 X Y	9 i j k m	
	Z	n p q r	
		A B C D	
		E F G	
		H J K L	
		M N P	
		Q R S T	
		U V W	
		10 X Y	
		Z	

$$NCHI = 3$$

$$NS = 2 * NCHI + 2 = 8$$

$$NS2 = 2 * NS = 16$$

Figure 5. Original Matrix

<p> A A B B C C D D A B B J C Q D E F F G G H E K F R G H L J M K N L P H S J W K C L M T N U P V M X N d P <u>5, 6, and 8</u> Q Y R Z S a T b Q e R i S n T U f V g W h U j V p W X k Y m X q Y Z r Z </p>	<p> A E J Q A E J Q B F K R B F K R C G L S W c C G L S W c D H M T X d D H M T X d N U Y e i n N U Y e i n P V Z f j p P V Z f j p a g k q a g k q <u>2 and 3</u> b h m r b h m r </p>	<p> A B C D A B C D E F G H E F G H J K L M N P J K L M N P Q R S T U V Q R S T U V W X Y Z a b W X Y Z a b c d e f g h c d e f g h i j k m i j k m <u>7 and 9</u> n p q r n p q r </p>
<p> NCHI = 3 NS = 2*NCHI + 2 = 8 NS2 = 2*NS = 16 </p>	<p> A B C D E F G H J K L M N P Q R S T U V W X Y Z a b c d e f g h i j k m <u>1</u> </p>	<p> A B C D E F G H J K L M N P Q R S T U V W X Y Z a b c d e f g h i j k m n p q r </p>
		<p> A B C D E F G H J K L M N P Q R S T U V W X Y Z <u>10</u> </p>

Figure 6. Reordered Matrix

ALBAND (8, ns2)

(subblocks 5, 6, and 8)

A	A	E	E	H	H	M	M	Q	Q	U	U	X	X	Z	Z
A	E	F	K	L	S	T	X	Y	e	f	j	k	q	r	
B	B	F	F	J	J	N	N	R	R	V	V	Y	Y		
B	J	G	R	M	W	U	d	Z	i	g	p	m			
C	C	G	G	K	K	P	P	S	S	W	W				
C	Q	H		N	C	V		a	n	h					
D	D			L	L			T	T						
D				P				b							

Figure 7. Storage of ALBAND.

OFDIAG (ns, 7, 5)

subblocks 2 3 4 and 4

OFDIAG (*, *, 1)

A	E	J	Q
B	F	K	R
C	G	L	S
D	H	M	T
N	U	Y	e
P	V	Z	f
a	g	k	q
b	h	m	r

OFDIAG (*, *, 2)

A	B	C	D
E	F	G	H
J	K	L	M
Q	R	S	T
W	X	Y	Z
c	d	e	f
i	j	k	m
n	p	q	r

OFDIAG (*, *, 3)

A	E	J	Q
B	F	K	R
C	G	L	S
D	H	M	T
N	U	Y	e
P	V	Z	f
a	g	k	q
b	h	m	r

OFDIAG (*, *, 4)

A	B	C	D
E	F	G	H
J	K	L	M
Q	R	S	T
W	X	Y	Z
c	d	e	f
i	j	k	m
n	p	q	r

OFDIAG (*, *, 5)

A	B	C	D
E	F	G	H
J	K	L	M
Q	R	S	T
W	X	Y	Z
c	d	e	f
i	j	k	m
n	p	q	r

Figure 8. Storage in OFDIAG

DTR $(ns^2*(ns^2+1)/2)$ Stored linearly by rows. Subblocks 1, 4, and 10.

A	B	C	D	E	F	G	H	A	B	C	D
	J	K	L	M	N	P	Q	E	F	G	H
		R	S	T	U	V	W	I	K	L	M
			X	Y	Z	a	b	Q	R	S	T
				c	d	e	f		W	X	Y
					g	h	i		c	d	e
						j	k	4		i	j
1							m			n	p
								A	B	C	D
									E	F	G
										H	J
											K
											L
											M
											N
											P
											Q
								10		R	S
										U	V
											W
											X
											Y
											Z

Figure 9. Storage in DTR