EB

MET O 11 TECHNICAL NOTE NO: 78

EFFICIENT USE OF DIRECT METHODS IN THE MESOSCALE MODEL

by

M C TAPP

CONTENTS

§ 1. INTRODUCTION

This note describes the incorporation into the Mesoscale model of a direct method for solving the set of Helmholtz equations

$$-\nabla_H^2\phi + \lambda_K\phi = F_K, \quad K=1,2\ldots L. \qquad (1)$$

In equation (1) ∇_H^2 is the horizontal Laplacian, $\lambda_K > 0$ is a constant and $F_K = F_K(x,y)$ is a field defined over a rectangular domain. The set of equations (1) arise from a semi-implicit scheme for numerically integrating the equations of motion. For a description of the scheme see Tapp and White (1976). If the model has $L+1$ levels then L equations of the type (1) have to be solved every timestep. It has been demonstrated (Tapp 1976) that direct methods can compare very favourably with iterative methods in the solution of such equations. However it became apparent that direct methods afford other programming advantages (most importantly core requirements) which are discussed in this note. The novel approach adopted here is thought to be of interest in any situation requiring the solution to the set of equations (1), especially when L is large. Direct methods also offer the advantage of producing a solution even when λ_K is small where iterative methods would otherwise be slow to converge.

§ 2. INCORPORATION OF DIRECT METHODS AND I/O STRATEGY

The semi-implicit scheme for numerically integrating the Mesoscale model requires the solution of the set of equations (1) every timestep. The existing method adopted to solve these equations has been the alternating direction implicit (hereafter ADI) method of solution (Peaceman and Rachford 1955). ADI is generally considered to be one of the most efficient iterative schemes, especially when (as in the Mesoscale model) the boundary conditions for (1) are of the Neumann or gradient type.

Despite the efficiency of the ADI method one drawback is the large amount of core required to store the right hand sides appearing in (1) (even in single precision). Thus for the current British Isles model about 131 k bytes of program are devoted entirely to this. Early attempts to remedy this by writing the appropriate fields to disk proved unsuccessful due to the excessive I/O

required. Separate work being done at about the same time indicated that a direct method might be more efficient at solving the set of equations (1) in terms of accuracy. However it was soon realised that, by coding the program in a certain way, the appropriate fields could be written off to disk and retrieved without penalty in overall efficiency.

To solve equations of the type (1) by direct methods, the reader is referred to Tapp (1976), but for completeness a brief outline is included here. For each Helmholtz equation on a rectangular grid of n rows and m columns, a matrix equation of the form

$$\underset{\approx}{Y}\,\underset{\approx}{\Phi} + \underset{\approx}{\Phi}\,\underset{\approx}{X} + \lambda_K \underset{\approx}{\Phi} = \underset{\approx}{F}_K \qquad (2)$$

can be written. The square tri-diagonal matrices $\underset{\approx}{Y}$ and $\underset{\approx}{X}$ represent the finite difference operators of $-\frac{\partial^2}{\partial y^2}$ and $-\frac{\partial^2}{\partial x^2}$ respectively, and $\underset{\approx}{\Phi}$ is the solution matrix of order n x m. $\underset{\approx}{F}_K$ is simply the matrix of right hand sides. Since λ_K is a constant over the rectangular domain, this matrix equation is especially simple. A matrix $\underset{\approx}{E}$ is sought such that

$$\underset{\approx}{E}^{-1}\,\underset{\approx}{X}\,\underset{\approx}{E} = \underset{\approx}{M},$$

where $\underset{\approx}{M}$ is a diagonal matrix with elements $(\mu_i, i = 1, 2, \ldots m)$ equal to the eigenvalues of $\underset{\approx}{X}$. Thus from (2)

$$\underset{\approx}{Y}\,\underset{\approx}{\Phi}\,\underset{\approx}{E} + \underset{\approx}{\Phi}\,\underset{\approx}{E}\,\underset{\approx}{M} + \lambda_K \underset{\approx}{\Phi}\,\underset{\approx}{E} = \underset{\approx}{F}_K\,\underset{\approx}{E} \qquad (3)$$

Putting $\underset{\approx}{\zeta} = \underset{\approx}{\Phi}\,\underset{\approx}{E}$, $\underset{\approx}{S}_K = \underset{\approx}{F}_K\,\underset{\approx}{E}$, (3) reduces to the system

$$\left(\underset{\approx}{Y} + (\lambda_K + \mu_i)\underset{\approx}{I}\right) K_i\left(\underset{\approx}{\zeta}\right) = K_i\left(\underset{\approx}{S}_K\right), \quad i = 1, 2, \ldots m, \qquad (4)$$

where $K_i\left(\underset{\approx}{B}\right)$ denotes the ith column of the matrix $\underset{\approx}{B}$. Once the solution for $\underset{\approx}{\zeta}$ has been found, the final answer may be recovered from

$$\underset{\approx}{\Phi} = \underset{\approx}{\zeta}\,\underset{\approx}{E}^{-1}.$$

Each of the equations (4) now depends only on the y co-ordinate. The solution vector $K_i\left(\underset{\approx}{\zeta}\right)$ and right hand side vector $K_i\left(\underset{\approx}{S}_K\right)$ are of dimension n. With L Helmholtz equations to solve, the problem can be reduced to solving L x m independent tri-diagonal equations of the type (4). Since $\lambda_K > 0$ and $\mu_i \geq 0$ the matrix

$$\underset{\approx}{Y} + (\lambda_K + \mu_i)\underset{\approx}{I}$$

is diagonally dominant and the standard tri-diagonal algorithm given by Varga

(1962, p. 195) may be used to solve (4). The method, which is a special adaption of Gaussian elimination, is often referred to in the literature as the double sweep method. For the equation

$$-\zeta_{j+1} + (2+\sigma)\zeta_j - \zeta_{j-1} = S_j, \quad j = 1, 2, \ldots m, \tag{5}$$

with boundary conditions $\zeta_0 = \zeta_1$, $\zeta_{n+1} = \zeta_n$ (typical Neumann conditions) and $\sigma = \lambda_k + \mu_i$, the solution may be written as (Richtmyer and Morton 1967 p 200)

$$\zeta_j = \alpha_j \zeta_{j+1} + \beta_j, \quad j = n-1, n-2, \ldots 1, \tag{6}$$

where

$$\alpha_j = \frac{1}{2+\sigma-\alpha_{j-1}}, \quad \beta_j = (S_j + \beta_{j-1})\alpha_j, \quad j = 2, \ldots n-1. \tag{7}$$

The boundary conditions give

$$\alpha_1 = \frac{1}{1+\sigma}, \quad \beta_1 = S_1 \alpha_1,$$

and (8a, b)

$$\zeta_n = (S_n + \beta_{n-1}) / (1 + \sigma - \alpha_{n-1}).$$

Thus from the boundary condition which gives α_1 and β_1 in (8a), the sequences $\{\alpha_j\}$ and $\{\beta_j\}$ may be computed inductively for increasing j. The boundary condition (8b) also gives the solution for $j = n$. The rest of the solution may then be found inductively from (6) in the order of decreasing j. The method is very efficient and can be shown to be absolutely stable with respect to round off error. It is apparent that the solution can be developed with the sweeps reversed, that is

$$\zeta_j = \alpha_j \zeta_{j-1} + \beta_j, \quad j = 2, 3, \ldots n, \tag{9}$$

where

$$\alpha_j = \frac{1}{2+\sigma-\alpha_{j+1}}, \quad \beta_j = (S_j + \beta_{j+1})\alpha_j, \quad j = n-1, \ldots 2. \tag{10}$$

Whichever method is adopted, clearly the calculation of the α_j's and β_j's or the final solution ζ_j depend only on the immediately preceding calculation. It is precisely this property of the double sweep method that enables the Helmholtz fields to be written to disk, since only one row of $\underset{\approx}{\zeta}$ and one row of α's and β's per Helmholtz equation are ever required at any one time.

For the Mesoscale model only a limited number of rows (three) of data are in main storage simultaneously for a particular calculation. It is thus convenient to order the solution of the set of equations (4) for each k so that they are all solved together, row by row. Instead of considering horizontal arrays of data $\underset{\approx}{F}_K \ (K=1,2,\ldots L)$ as in (2), then since

$$\underset{\approx}{F}_K = \{ \underset{\approx}{F}_{ijK} \},$$

where i refers to columns, j to rows and K to the different Helmholtz equations, it is possible to reorder so that

$$\underset{\approx}{C}_j = \{ C_{iK} \}_j = \{ F_{ijK} \},$$

for a fixed row j. The matrix $\underset{\approx}{C}_j$ is then of order $L \times m$ (L rows and m columns) and contains all the F_{ijK} values for a particular row j of the set of equations (1). The elements of $\underset{\approx}{C}_j$ are ordered so that each row corresponds to a Helmholtz equation. Hence the right hand side of (4), for each row can be computed as the matrix product

$$\underset{\approx}{d}_j = \{ d_{iK} \}_j = \underset{\approx}{C}_j \underset{\approx}{E} . \qquad (11)$$

For each d_{iK}, using the recurrence relations for the double sweep method, a pair of coefficients α_{iK} and β_{iK} can be calculated. Each line calculation of the Mesoscale model gives the right hand sides F_{ijK} for all i and K but only for a particular row j. However this is all that is necessary to

(a) decouple the equations in the x-direction according to (11) and

(b) to do the elimination process in the y-direction for the next row of α's and β's.

Each elimination is performed per row for all the $L \times m$ equations together and the resulting buffer containing the α and β values can be written to a dataset each time. The solution for the final row may be developed in core since this requires storing only one row of answers per Helmholtz equation.

The answers to the equations (4) ζ_{ijk} and the final solution can be ordered as above, so that

$$\underset{\approx}{\Phi}_j' = \{\Phi_{ijk}\}, \quad \underset{\approx}{\zeta}_j' = \{\zeta_{ijk}\}$$

for fixed j. The matrices $\underset{\approx}{\Phi}_j'$ and $\underset{\approx}{\zeta}_j'$ are of order $L \times m$. The solution field $\underset{\approx}{\Phi}_j'$ for each row can be found from the matrix product

$$\underset{\approx}{\Phi}_j' = \underset{\approx}{\zeta}_j' \underset{\approx}{E}^{-1} . \qquad (12)$$

The matrix products (11, 12) may be computed using fast Fourier transform techniques.

Because of the nature of the double sweep method, the α, β records must be accessed in reverse order to the order in which they are written up. Each record read down can be used to compute the next row of answers ($\underset{\approx}{\zeta}_j'$) overwriting those currently held in the buffer. The strategy adopted here for solving a large number of equations simultaneously is similar to that adopted by Wang (1974) in devising the block-ADI method for virtual storage machines.

The Mesoscale model uses a main work data set (usually residing on the fixed head disk) which stores data temporarily while row calculations are being performed. To incorporate the above direct method of solution this dataset, as well as the Helmholtz dataset, has to be scanned in alternating directions each timestep. This approach is considered quite novel since the usual approach is to process the data in the same direction each timestep. The new I/O system is displayed in figures 1-3. Note that apart from the first timestep, three writes and three reads of data (to the fixed head disk) are saved every time-step because of the alternating direction of scan.

§ 3. PROGRAMMING CONSIDERATIONS

In the previous section it has been tacitly assumed that the datasets used by the program can be processed sequentially forwards and backwards. This is easily achieved using the elementary access technique called execute channel program (EXCP). Since the records involved are large, the starting addresses for each, all begin on a new track. Access to a particular record can be made using the cylinder (CC) and head (HH) or track number. Once the datasets are established, the CCHH numbers can be stored in the program in an array affording a unique correspondence between elements of that array and the records stored on a disk. Since with the EXCP method, access to the records need not be sequential, they may be processed in any convenient way. For the purposes of solving the set of equations (1) only a sequential forwards and backwards processing is required. Clearly for the main work dataset on the fixed head disk this results in no loss of efficiency at all - since the heads do not move. In fact use of this dataset is improved since at the turnround stage between timesteps no reading or writing is taking place. The Helmholtz dataset is placed on system space (3330 disk) since this uses a separate channel - I/O to the two datasets can thus run concurrently. Each record of the Helmholtz dataset contains a row of α's and β's (stored in double precision) which conveniently fits on one track. One problem which may arise for this dataset is that of excessive head movement in reading and writing in both directions. However, to date, no difficulties have been encountered.

For the Mesoscale model, approximately 46 k bytes of program are devoted to storing and solving the set of equations (1) - a saving of about 90 k over the ADI method. The maximum value of $L \times M$ that can be catered for is 640 while there is now no restriction (in principle) on the number of rows.

Defining north to be in the positive y-direction, under the old system the dataset was always processed from north to south each timestep. The procedure used in the new scheme, described here, is to scan from north to south on the even timesteps and south to north on the odd timesteps. In the

odd timestep scan, the buffers appear in the 'wrong' order for the routines used in the even timestep calculation. One can either write different routines to do the odd timestep calculations, or transform the data after processing so that the existing routines can still be used. The latter approach has been adopted in the Mesoscale model. The grid system is displayed in figure 4. Thus at each timestep, the co-ordinate transformation

$$\begin{aligned} x' &= x \\ y' &= -y \\ z' &= z \end{aligned} \qquad (13)$$

is applied to prepare the data for use at the next timestep. In the equations of motion scalars (density, pressure, temperature etc) are unaffected by such a transformation, while the velocity $\underset{\sim}{V}$ obeys

$$(u', v', w') = (u, -v, w)$$

in component form. The coriolis term must also be transformed as

$$f' = -f$$

since the vector product $f\underset{\sim}{k} \wedge \underset{\sim}{V}$ is determined by a different rule in each co-ordinate system. Since the v's are staggered in the y-direction with respect to the grid, they must be moved around between the buffers to appear in the right position for the routines each timestep. This presents a slight technical problem but it does not prove expensive in terms of CPU time. In fact it takes about half a percent of the computing time but is more than offset by the efficient program that results.

The direct method has been incorporated in the cumulus version of the Mesoscale model. (The appendix gives the routines and what they do for reference). As a rigorous test of the system, the method was programmed as above, and also the alternate way keeping the right hand side fields in core and using the old I/O routines. Differences between the forecasts highlighted a special problem that could arise. Since the same routines are used for the two different grid systems, data appears in reverse order in the buffers (in the y-direction sense). This means that although the routines perform the same operations on the numbers (and should in theory get the same result), the order of operations may change. For

example, suppose at a particular grid point calculation three v-component velocities for the lines $n-1$, n and $n+1$ have to be added together. Normally one would write

$$V = V_{n-1} + V_n + V_{n+1} . \qquad (14)$$

However, in Fortran this would be added as though

$$V = (V_{n-1} + V_n) + V_{n+1}$$

In the transformed grid (apart from the sign of V which is automatically taken care of) the variables would appear in reverse order, so that

$$V = V_{n+1} + V_n + V_{n-1} = (V_{n+1} + V_n) + V_{n-1} . \qquad (15)$$

In floating point arithmetic (14) may differ (slightly) from (15). Thus unwittingly the programmer may introduce different rounding errors between each timestep. However, in floating point arithmetic, the result of adding or multiplying two numbers together doesn't depend on their order, so that

$$A+B = B+A \quad , \quad AB = BA . \qquad (16)$$

Hence (14) may be reprogrammed to give

$$V = (V_{n-1} + V_{n+1}) + V_n , \qquad (17)$$

which is now invariant (apart from sign) with respect to the grid transform. In this way, differing rounding errors on the two grids can be eliminated in the dynamical calculations.

The only remaining possible source of error is the solution of the tri-diagonal systems, which must be evaluated in opposite directions each successive timestep. Tested on the same data, the two methods of sweeping show differences only in the last two hex digits of the double precision answer. Since the final results of the Helmholtz calculations are required only in single precision, this error can never feed back into the dynamics.

To summarise, by using the simple laws of floating point arithmetic (16), the Fortran programmer can ensure that the results will be consistent in both co-ordinate systems. Since only the y-axis is effectively transformed only those operations (such as derivatives or averaging) involving that co-ordinate need be scrutinised. In practice this usually requires inserting extra brackets to force the compiler to add the terms in the desired way.

The only remaining problem is what happens at restart points. Since the time-stepping scheme is leapfrog, two datasets at consecutive time levels are needed at each restart point. To fit in with the current scheme, the writeup data is written to disk in the normal way (during an even timestep), but the restart data (at the odd timestep) is written up in reverse order. Since the restart dataset is only there to enable the integration to continue, it is of no consequence that the data is 'jumbled'. It is a straightforward matter to 'unscramble' the appropriate data when needed.

Despite the extra I/O (increased by a factor of about 2.5) the new version of the model suffers practically no loss of efficiency. Measured as

$$\frac{\text{CPU TIME}}{\text{ELAPSED TIME}} \times 100\%$$

it is normally around the 95% mark. Experience in running has shown no practical disadvantages - only the gain in superior program design and flexibility.

REFERENCES

Peaceman, D.W. and Rachford, H.H. 1955 'The numerical solution of parabolic and elliptic differential equations'. J. Soc. Indust. App. Maths., 3, pp.28-41.

Richtmyer, D.W. and Morton, K.W. 1967 'Difference methods for Initial Value Problems'. John Wiley.

Tapp, M.C. 1976 'A direct method for the solution of Helmholtz-type equations'. Met O 11 Tech. Note No: 71.

Tapp, M.C. and White, P.W. 1976 'A non-hydrostatic mesoscale model'. Quart. J. R. Met. Soc., 102, pp.277-296.

Varga, R.S. 1962 'Matrix Iterative Analysis'. Prentice Hall.

Wang, H.H. 1974 'An ADI procedure suitable for virtual storage systems'. IBM Palo Alto Scientific Center Tech. Report No: G320-3322. JAN..

APPENDIX

This appendix lists the main I/O routines and those involved in the solution of the Helmholtz equations. A basic flow diagram is given in figure 5. Routines that calculate the main dynamics are called from LINE but are excluded from figure 5 for clarity.

BEGIN : Sets up main work dataset for a forecast and writes initial or restart data to it.

STEP : Reads down sufficient data to perform row calculations and writes the computed results back up. The dataset is processed a row at a time until the calculation is complete. Direction of scanning is switched every timestep.

MOVE : Before both time level buffers are written up the v-component of velocities is transformed. This necessitates moving the v's between the line buffers and a temporary holding buffer.

LINE : Calls various routines (not included in figure 5) to perform line calculations.

DECPLE : Computes a row of right hand sides for each Helmholtz equation. After transforming the rows (call to UNCPLE) the coefficients for the elimination process are calculated and written to disk (call to WRTE).

UNCPLE : Transforms by row the right hand sides of the Helmholtz equation.

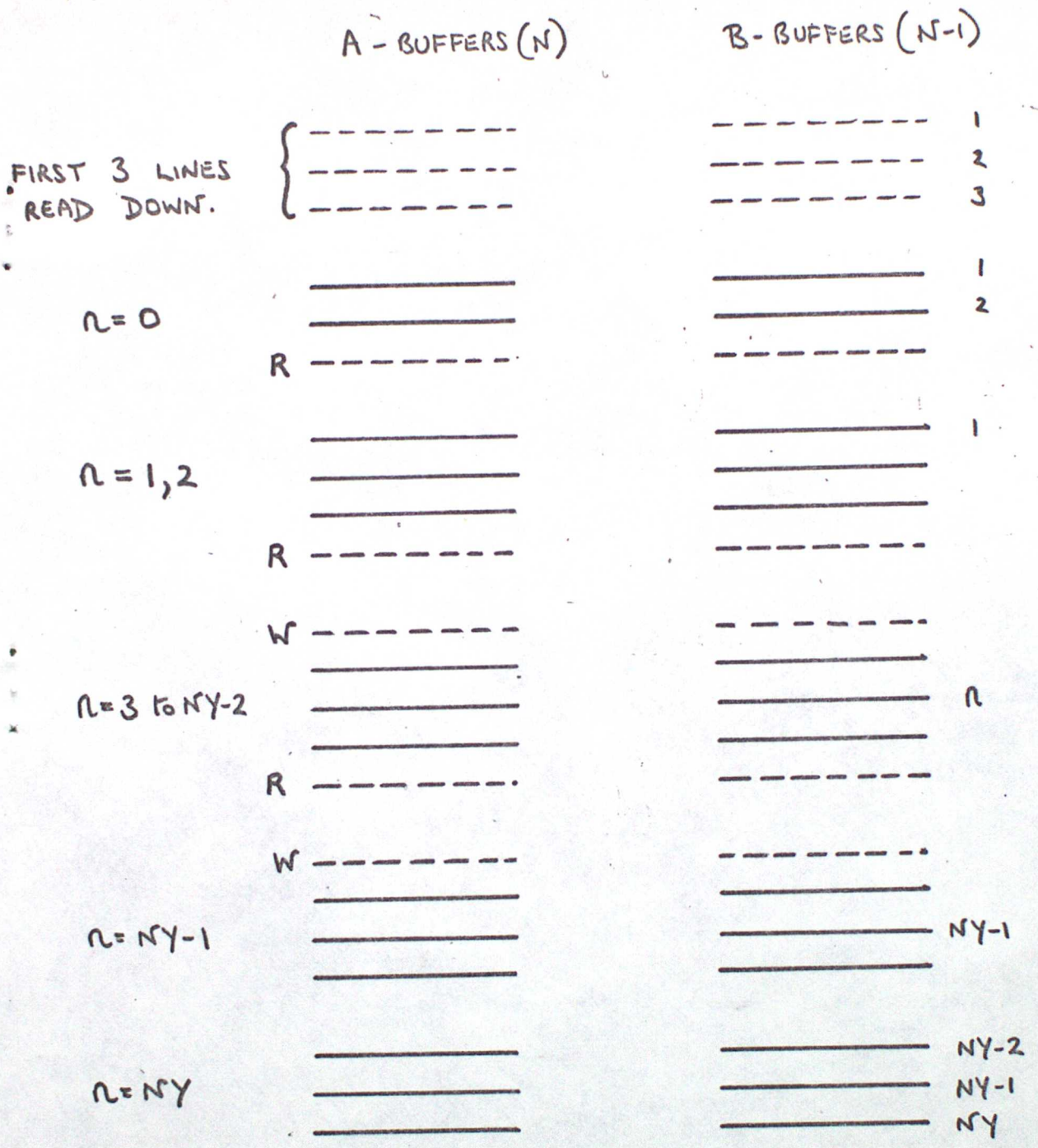
FFT2 : Computes a basic sine transform.

FINAL : Reads down the α, β coefficients (call to READ) to complete the solution of the tri-diagonal equations. After transforming the rows (call to RECPLE), the results are used to complete the dynamical calculations of the previous timestep.

RECPLE : Effectively performs the inverse transform to UNCPLE to recover the solution.

WRTE : Writes the Helmholtz records (α's and β's) to disk.

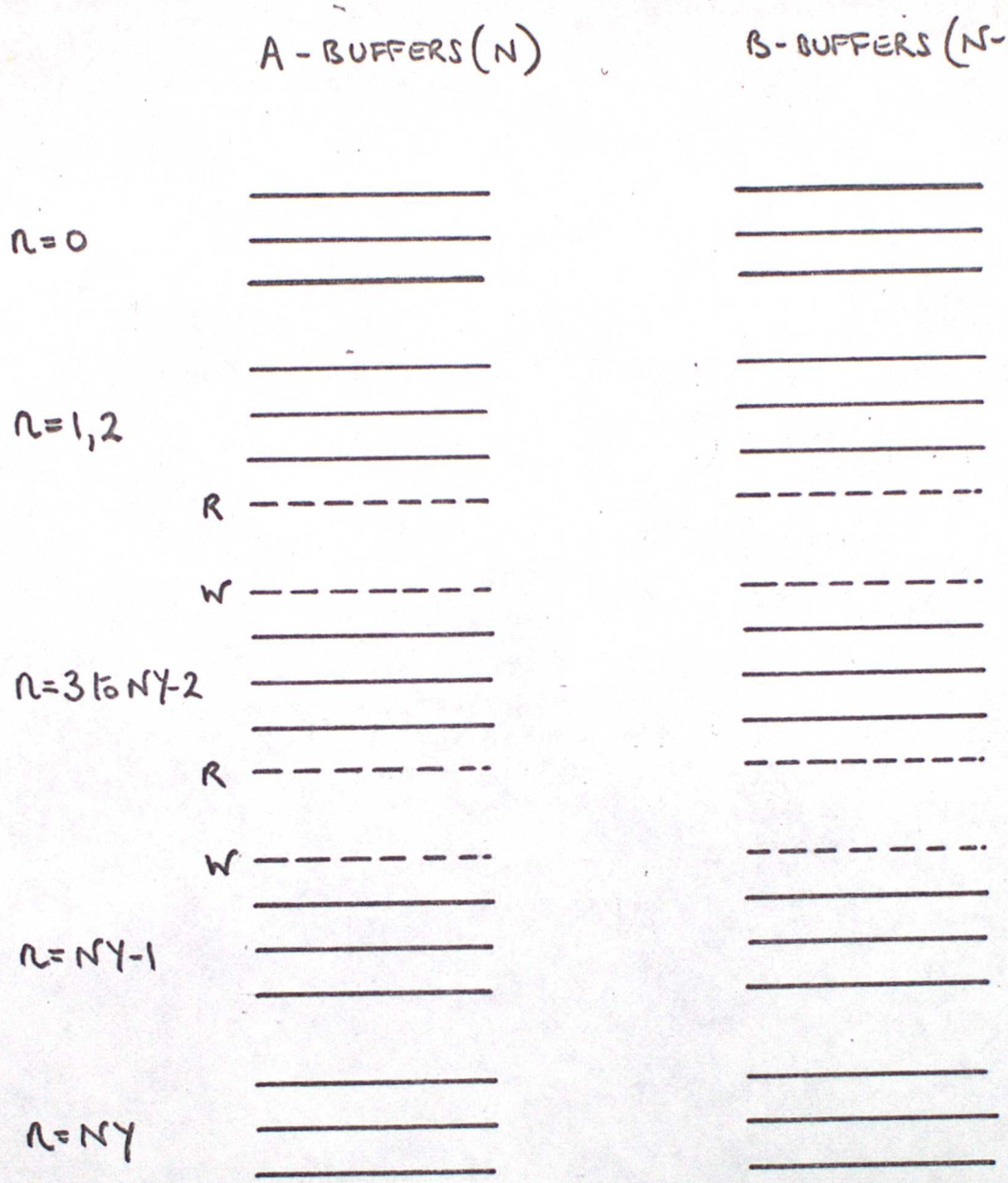
READ : Reads down the Helmholtz records.



KEY

R = READING
W = WRITING
N = TIME LEVEL
n = ROW COUNT
NY = TOTAL NUMBER OF ROWS

FIGURE 1. I/O SYSTEM FOR MAIN WORK DATASET (FIRST TIMESTEP)



FOR KEY SEE FIGURE 1.

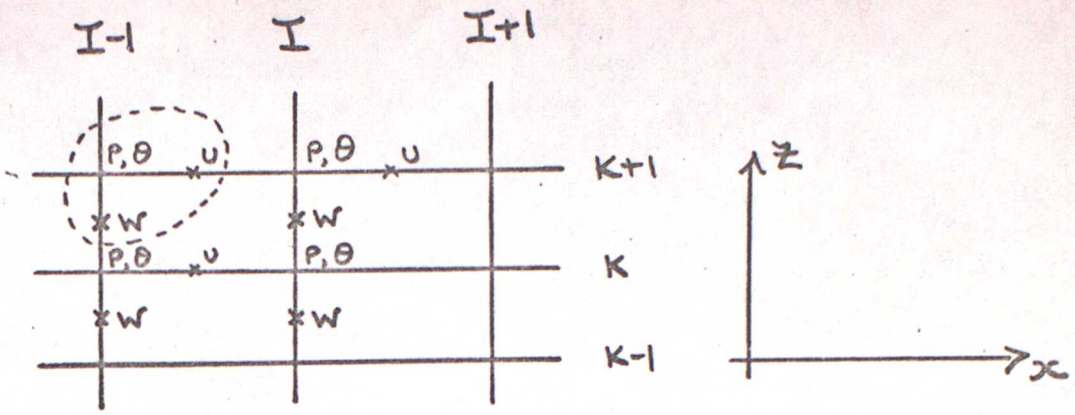
FIGURE 2. I/O SYSTEM FOR MAIN WORK DATASET (AFTER THE FIRST TIMESTEP)

	READ ROUTINE	WRTE ROUTINE
$\ell = 0, 1$	R --------	NOT CALLED
$\ell = 2$ to NY-4	R --------	W --------
ℓ = NY-3	NOT CALLED	W --------
ℓ = NY-2	NOT CALLED	----------
		LAST RECORD COMPUTED AND KEPT IN CORE.
ℓ = NY-1	NOT CALLED	LAST LINE OF SOLUTION DEVELOPED AND STORED IN BUFFER.

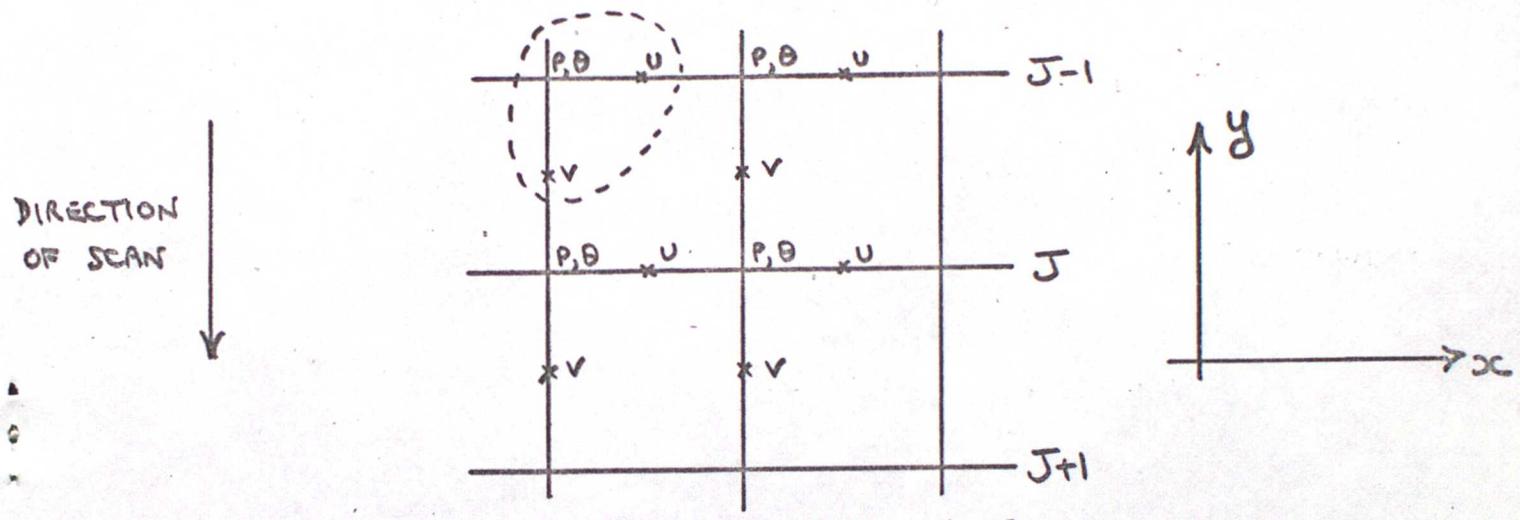
NO READING TAKES PLACE DURING THE INITIAL TIMESTEP

FOR KEY SEE FIGURE 1.

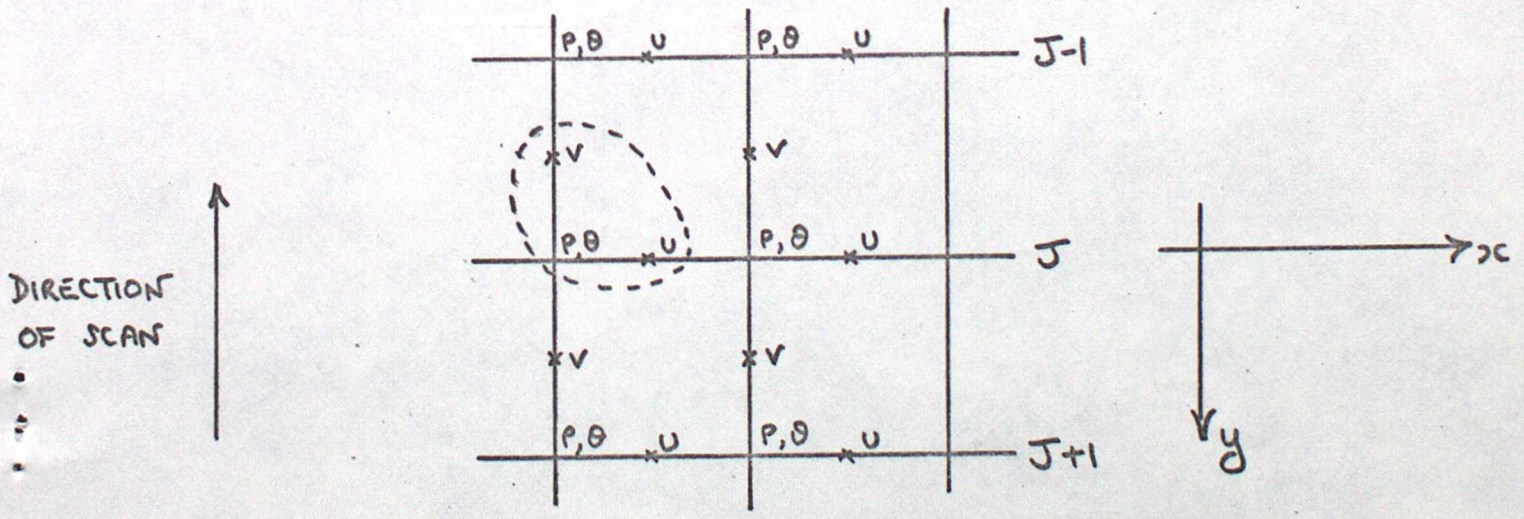
FIGURE 3. I/O SYSTEM FOR HELMHOLTZ DATASET



VERTICAL STRUCTURE (EITHER TIMESTEP)

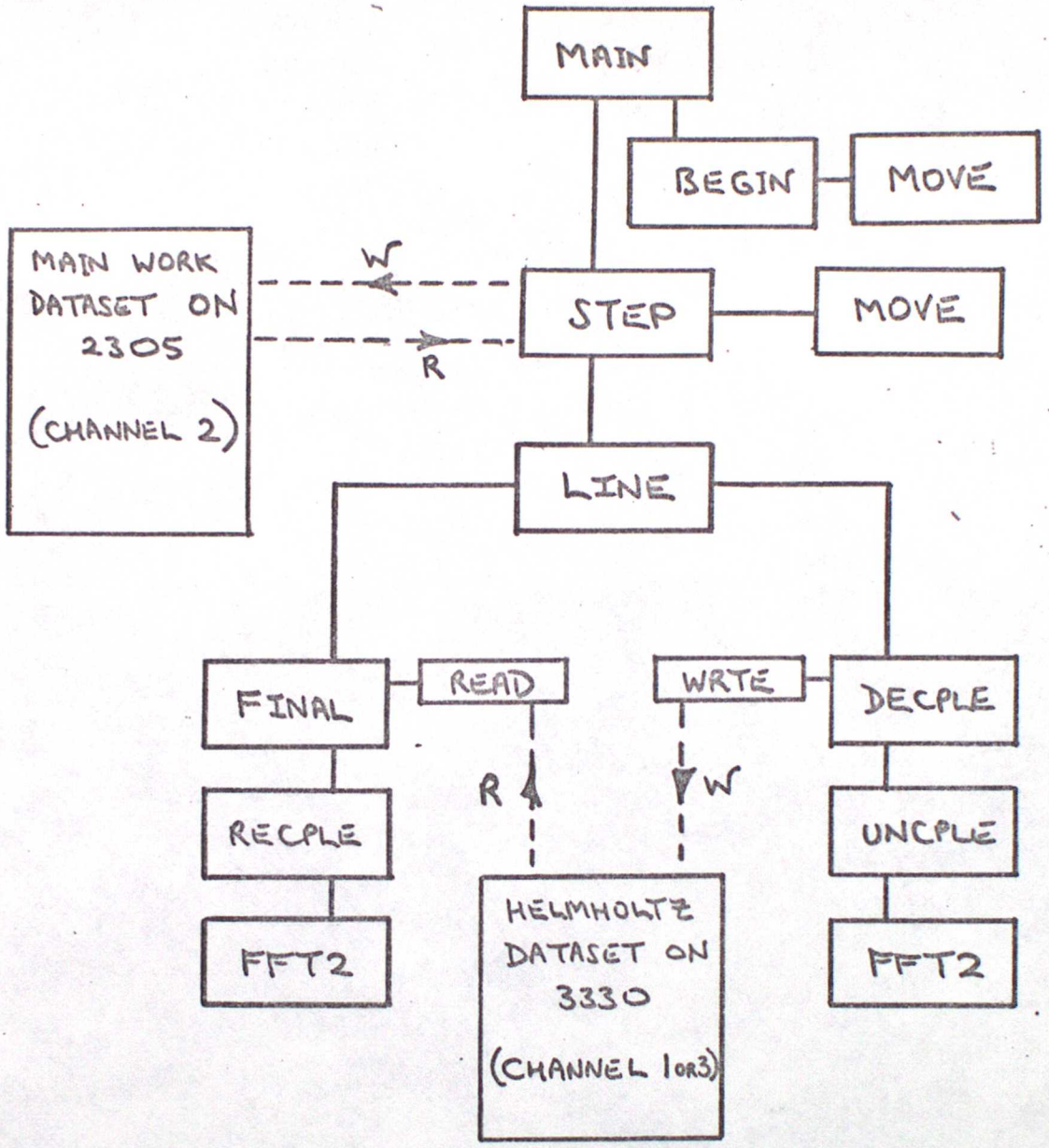


HORIZONTAL STRUCTURE
(EVEN TIMESTEP)



HORIZONTAL STRUCTURE
(ODD TIMESTEP)

FIGURE 4. GRID STRUCTURE FOR THE MESOSCALE MODEL.



- - - - DENOTES DATA FLOW
R = READING
W = WRITING

FIGURE 5 MESOSCALE MODEL FLOW DIAGRAM FOR I/O AND SOLUTION OF HELMHOLTZ EQUATIONS.