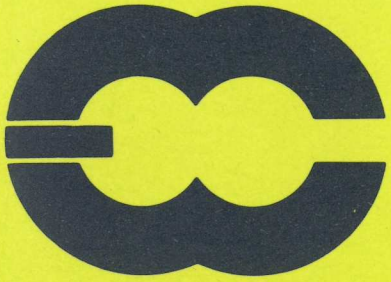


European Centre
for Medium Range
Weather Forecasts

Users Guide for the G.F.D.L. Model

November 1976

Centre Européen pour les Prévisions Météorologiques
a Moyen Terme

Europäisches Zentrum Für Mittelfristige Wettervorhersagen

USERS' GUIDE OF THE GFDL MODEL AT THE EUROPEAN CENTRE FOR MEDIUM RANGE WEATHER FORECASTS

A.P.M. Baede
D. Dent
A. Hollingsworth

Bracknell, November 1976

Introduction

At present ECMWF has at its disposal both the N24 and N48 9-layer versions of the GFDL model. This guide was written to simplify the introduction to and the use of the model, on both the CDC 6600 and the IBM 360/195. The documentation, kindly supplied to us by GFDL, is quite appropriate as an introduction, but a few parts were not well documented. These parts have been written by us and added to this guide.

Chapter 1 gives a survey of the documentation, available at present. The next two chapters contain the new documentation of both the horizontal diffusion scheme and the I/O scheme. Chapter 4 gives a survey of the NAMELIST INPUT parameters. The last chapter contains the instructions for using the model on both computers. Special instructions for the operators of the CDC 6600 are added in an appendix.

.../2

Chapter 1 - Available documentation

The following documentation is available :

1. The source text of:

   1.1 The N24 version on the CDC 6600
   1.2 The N48 version on the CDC 6600
   1.3 The N24 version on the IBM 360/195
   1.4 The N48 version on the IBM 360/195

2. A "dictionary" of the N48 CDC-version, containing an alphabetic list of all variables, externals, common blocks and file names, their relocations, references and subroutines where they are defined.

3. A written documentation, prepared and kindly made available to us by GFDL, of the following subjects:

   3.1 Basic equations and the finite difference equations
   3.2 Subroutines and Control block
   3.3 Equations for u,v, T and r, except sub-grid scale convection and macro-scale condensation process
   3.4 Vertical fluxes of momentum and moisture in the planetary boundary layer and the free atmosphere
   3.5 Hydrology at the Earth's surface
   3.6 Sub-grid scale convection and macro-scale condensation process
   3.7 Radiation, part I: Solar radiation
   3.8 Radiation, Part II:Long wave radiation
   3.9 Integral quantities calculated by the program
   3.10 Budget checks.

4. A written documentation, prepared by us and added as chapter 2 and 3 to this guide of the following subjects:

   4.1 The horizontal diffusion
   4.2 The I/O scheme

5. A survey of input parameters to be read in via NAMELIST INPUT. This is chapter 4 of this guide.

6. Users instructions, prepared by us and added as chapter 5 to this guide, for the use of :

   6.1 The N24 and N48 version on the CDC 6600
   6.2 The N24 version on the IBM 360/195.
   6.3 The N48 version on the IBM 360/195.

7. Operator instructions, prepared by us and added as an appendix to this guide for the use of both versions on the CDC 6600.

.../4

Chapter 2 - The horizontal diffusion

1. Outline

This note describes the horizontal diffusion in the GFDL model. The quantities H^F_u, H^F_v, H^F_T and H^F_R, representing the diffusive change of the wind components, the heat and the relative humidity, in flux form, are calculated in the subroutine HORZDF. The horizontal diffusion on the wind components is performed on σ -surfaces, whereas the diffusion of heat and moisture takes place in p-surfaces to avoid an unrealistic pushing of heat and moisture on to the top of mountains.
To avoid instability of the finite difference scheme, the diffusion terms are evaluated at the previous time step. The subroutine HORZDF is called from MAIN, after the radiation but before the vertical diffusion. The horizontal diffusion of T and R on pressure surfaces requires linear interpolation between σ-surfaces which is performed in the subroutine TR1 + TSTT and TR2 + TST2 for the central points of the neighbour boxes and the central box respectively. So the general outline of the subroutines is as follows:

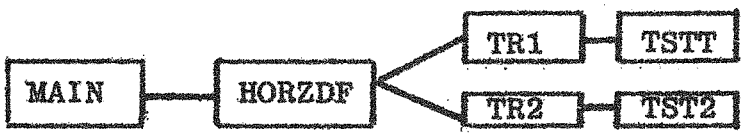


FIG. 1

2. Horizontal diffusion of wind components

2.1 Basic equations

The equations for non-linear horizontal diffusion of the wind components (in flux form), as used in the GFDL model are:

$$H^{F}_{u} = (2k_o^2\ a^2\Delta\lambda^2\ \cos^2\ \phi)\quad \frac{\partial}{a\cos\ \phi\partial\lambda}\ (p_*D_t|D|\ ) + 2k_o^2\ a^2\Delta\phi^2$$

$$\frac{1}{a\cos^2\ \phi}\quad \frac{\partial}{\partial\phi}\quad (p^*D_s|\ D|) \qquad (1)$$

$$H^{F}_{v} = (2k_o^2\ a^2\Delta\lambda^2\cos^2\ \phi\ )\ \frac{\partial}{a\cos\ \phi\partial\lambda}\ (p_*D_s|D|) - 2k_o^2\ a^2\Delta\phi^2$$

$$\frac{1}{a\cos^2\ \phi}\quad \frac{\partial}{\partial\phi}\quad (p^*D_t|D|) \qquad (2)$$

where D_t and D_s are the tension and shearing strains :

$$D_t = \frac{\partial\ u}{a\cos\phi\partial\lambda} - \frac{\cos\phi}{a}\quad \frac{\partial}{\partial\phi}\ (\frac{v}{\cos\phi}) = \frac{\partial u}{a\cos\phi\ \partial\lambda} - \frac{\partial\ v}{a\partial\phi} - \frac{tg\phi}{a}v \qquad (3)$$

$$D_s = \frac{\partial v}{a\cos\phi\partial\lambda} + \frac{\cos\ \phi}{a}\quad \frac{\partial}{\partial\phi}\ (\frac{u}{\cos\phi}) = \frac{\partial\ v}{a\cos\ \phi\partial\lambda} + \frac{\partial u}{a\partial\phi} + \frac{tg\ \phi}{a}v \qquad (4)$$

$$D = \sqrt{D_s^2 + D_t^2} \qquad (5)$$

$k_o = 0.125$

and the other symbols are self-explanatory.

.../6

2.2 Finite difference form of the equations

Using the box-method the derivatives in (1) and (2) are approximated as follows :

$$\frac{\partial}{a\cos\phi\,\partial\lambda}\,(p^*D_t|D|) = \frac{\delta_\lambda(p^*D_t|D|)}{a\cos\phi\,\Delta\lambda} \qquad (6)$$

$$\frac{\partial}{a\,\partial\phi}(p^*D_s|D|) = \frac{\delta_\phi(p^*D_s|D|)}{a\,\Delta\phi} \qquad (7)$$

with:

$$\delta_\lambda(p_*D_T|D|) = \left(\sum_E - \sum_W\right) w_\ell\,\overline{p_*}^{\lambda}\left(\frac{\delta_1 u}{a\cos\varphi_j\Delta\lambda} - \frac{\operatorname{tg}\varphi_0}{a}\,\overline{v}^{\lambda}\right)\cdot$$
$$\cdot\sqrt{\left(\frac{1}{a\cos\varphi_j\Delta\lambda}\,\delta_1 u - \frac{\operatorname{tg}\varphi_0}{a}\,\overline{v}^{1}\right)^2 + \left(\frac{1}{a\cos\varphi_j\Delta\lambda}\,\delta_1 v + \frac{\operatorname{tg}\varphi_0}{a}\,\overline{u}^{1}\right)^2} \qquad (8)$$

$$\delta_\varphi(p_*D_S|D|) = \left(\sum_N - \sum_S\right)\frac{\cos(\varphi_0 \pm \tfrac{1}{2}\Delta\varphi)}{\cos\varphi_0}\,w_\ell\,\overline{p_*}^{\varphi}\,\frac{\cos(\varphi_0 + \tfrac{1}{2}\Delta\varphi)}{a\,\Delta\varphi_0}\,\delta_\varphi\left(\frac{u}{\cos\varphi}\right)\cdot$$
$$\cdot\sqrt{\frac{\cos^2(\varphi_j \pm \tfrac{1}{2}\Delta\varphi)}{a^2\Delta\varphi_0^2}\left\{\delta_\varphi^2\left(\frac{u}{\cos\varphi}\right) + \delta_\varphi^2\left(\frac{v}{\cos\varphi}\right)\right\}} \qquad (9)$$

Here the suffix j refers to row j, at which we are, whereas 1 refers to the neighbour boxes (o refers to the central point)(see fig.2).
It should be noted that the zonal variation of a quantity at a N or S interface and the meridional variation at an E or W interface are neglected, although the terms $\frac{\operatorname{tg}\phi}{a}v$ and $\frac{\operatorname{tg}\phi}{a}u$,which in fact represent part of the meridional variation, are retained . It should further be noted that the first factor on the RHS of eq. (9) is due to averaging over the box and is close to unity anyhow.

.../7

2.3 Implementation

The subroutine HORZDF starts with the calculation of a number of constants at every first point of a row (LARRAY (2) =.FALSE.) Referring for details to the codes dictionary, the following trigonometric constants are defined (the suffix or parameter j referring to the number of the row : j is equivalent to element 32 of COMMON/CTBLK/):

COSJ (J) = $\cos\phi_j$

RCOSJ(J) = $\frac{1}{\cos\ \phi_j}$

COSJN (J) = $\cos\ (\phi_j + \frac{1}{2}\Delta\phi_j\ )$

COSJS (J) = $\cos\ (\phi_j - \frac{1}{2}\Delta\phi_j\ )$

The following constants are calculated :

TANA = $\frac{tg\phi_j}{2a}$

TANA2 = $\frac{tg\phi_j}{a}$

ACTHL = $\frac{1}{a\cos\ \phi_j\ \Delta\lambda_j}$

ATCSN = $\frac{\cos\ (\phi_j + \frac{1}{2}\Delta\phi_j)}{a\Delta\phi_j}$

ATCSS= $\frac{\cos\ (\phi_j - \frac{1}{2}\Delta\phi_j)}{a\Delta\phi_j}$

HK1 = $ko^2 a^2 \Delta\phi_j^2$

The last three constants are corrected with a factor 1.5 if we are at the row next to the pole.

.../8

$$HK2 = k_o^2 \, a^2 \cos^2 \phi_j \, \Delta^2 \lambda_j$$

$$HK3 = k_o^2 \, a \cos \phi_j \, \Delta \lambda_j$$

$$RCJNAN = \frac{\cos\left(\phi_j + \frac{1}{2}\Delta\phi_j\right)}{a\Delta\phi_j \cos\phi_{j-1}}$$

$$RCJJAN = \frac{\cos\left(\phi_j + \frac{1}{2}\Delta\phi_j\right)}{a\Delta\phi_j \cos\phi_j}$$

$$RCJJAS = \frac{\cos\left(\phi_j - \frac{1}{2}\Delta\phi_j\right)}{a\Delta\phi_j \cos\phi_j}$$

$$RCJSAS = \frac{\cos\left(\phi_j - \frac{1}{2}\Delta\phi_j\right)}{a\Delta\phi_j \cos\phi_{j+1}}$$

For every point the following quantities are defined :

$PST = p_* \, (0)$

$$RPSTM = \frac{1}{p_{*-}(0)}$$

in which parameter 0 refers to the central box and the suffix - refers to the previous time step.

The next 400-loop and the calls for TR1 and TR2 are part of the T and R diffusion and will be discussed later.

The final calculation takes place in the 100-loop for all kmax vertical layers. In the following suffix k refers to the σ-level. Parameters 0 to 6 refer to the central box and its neighbours according to the following scheme :

.../9

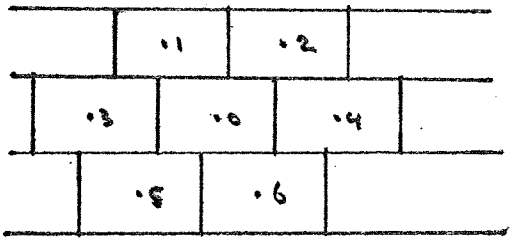


Fig. 2

First some σ- level dependent constants are calculated for the E-W direction. For the first point of the row the following west neighbour quantities are calculated:

$$DT3(K) = \frac{u_{k-(0)} - u_{k-(3)}}{a\cos\varphi_j \Delta\lambda_j} - \frac{\tan\varphi_j}{2a}\left[v_{k-(0)} + v_{k-(3)}\right] = \frac{\delta_2 u_k}{a\cos\varphi_j \Delta\lambda_j} - \frac{\tan\varphi_j}{a}\overline{v_k}^{\lambda} = (D_T)_k$$

$$DS3(K) = \frac{v_{k-(0)} - v_{k-(3)}}{a\cos\varphi_j \Delta\lambda_j} + \frac{\tan\varphi_j}{2a}\left[u_{k-(0)} + u_{k-(3)}\right] = \frac{\delta_2 v_k}{a\cos\varphi_j \Delta\lambda_j} + \frac{\tan\varphi_j}{a}\overline{u_k}^{\lambda} = (D_S)_k$$

$$SQD3(k) = \sqrt{DT3(k)^2 + DS3(k)^2} = \sqrt{(D_t^2 + D_s^2)_k}$$

$$PSDW3(k) = W_3 \cdot \left[p_{*-}(o) + p_{*-}(3)\right] \cdot \sqrt{DT3(k)^2 + DS3(k)^2} = 2W_3 \overline{p_*}^{\lambda} \sqrt{(D_t^2 + D_s^2)_k}$$

Analogously the following east neighbour quantities are calculated.

$$DT4(k) = \frac{\delta_2 u}{a\cos\varphi_j \Delta\lambda_j} - \frac{\tan\varphi_j}{a}\overline{v}^{\lambda} = (D_T)_k$$

$$DS4(k) = \frac{\delta_2 v}{a\cos\varphi_j \Delta\lambda_j} + \frac{\tan\varphi_j}{a}\overline{u}^{\lambda} = (D_S)_k$$

$$SQD4\ (k) = (D_t^2 + D_s^2)_k$$

$$PSDW4\ (k) = 2W_4 \overline{p_*}^{\lambda} (D_t^2 + D_2^2)_k$$

.../10

For all but the first points in the row all west neighbour quantities are set equal to the east neighbour quantities of the previous point and new east neighbour quantities are calculated. Next in the 10- and 11-loop quantities with regard to the N and S neighbours are calculated respectively. Neighbours with a weight smaller than 10^{-13} (SP (I) = . FALSE.) are neglected.

10-loop:

$$DT = \frac{\cos(\varphi_j + \frac{1}{2}\Delta\varphi_j)}{a\cos\varphi_j\,\Delta\varphi_j} V_{k-}(0) - \frac{\cos(\varphi_j + \frac{1}{2}\Delta\varphi_j)}{a\cos\varphi_{j-1}\,\Delta\varphi_j} V_{k-}(I) = \frac{\cos(\varphi_j + \frac{1}{2}\Delta\varphi_j)}{a\,\Delta\varphi_j} \delta_\varphi \left(\frac{v_k}{\cos\varphi}\right.$$

$$= (D_T)_k$$

$$DS = -\frac{\cos(\varphi_j + \frac{1}{2}\Delta\varphi_j)}{a\cos\varphi_j\,\Delta\varphi_j} \cdot u_{k-}(0) + \frac{\cos(\varphi_j + \frac{1}{2}\Delta\varphi_j)}{a\cos\varphi_{j-1}\,\Delta\varphi_j} u_{k-}(I) = \frac{\cos(\varphi_j + \frac{1}{2}\Delta\varphi_j)}{a\,\Delta\varphi_j} \delta_\varphi \left(\frac{u_k}{\cos\varphi}\right.$$

$$= (D_S)_k$$

$$PSD = w_z \cdot [p_{k-}(0) + p_{k-}(I)] \sqrt{DT^2 + DS^2} = 2 w_z \overline{p_*}^{\varphi} \sqrt{(D_T^2 + D_S^2)_k}$$

$$SUMN1 = \sum_N 2 w_z \overline{p_*}^{\varphi} (D_S)_k \sqrt{(D_S^2 + D_T^2)_k}$$

$$SUMN2 = \sum_N 2 w_z \overline{p_*}^{\varphi} (D_T)_k \sqrt{(D_S^2 + D_T^2)_k}$$

11-loop:
Analogous expressions for the south neighbours.
Then finally expressions for ${}_H F_u$ and ${}_H F_v$ are calculated.

.../11

3. Horizontal diffusion of temperature and humidity

Because R and T diffusion are exactly analogous only T diffusion is treated here.

3.1 The basic equation

The equation for non-linear horizontal diffusion of the temperature is :

$$nF_T = 2k_0^2 a^2 \Delta\lambda^2 \cos^2\varphi \frac{\partial}{a\cos\varphi\,\partial\lambda}\left[P_*\left(\frac{\partial T}{a\cos\varphi\,\partial\lambda}\right)_p \sqrt{D_s^2 + D_T^2}\right]$$

$$+ 2k_0^2 a^2 \Delta\varphi^2 \frac{\partial}{a\,\partial\varphi}\left[P_*\left(\frac{\partial T}{a\,\partial\varphi}\right)_p \sqrt{D_s^2 + D_T^2}\right] \qquad (10)$$

Here the T gradients, as mentioned before, are evaluated on pressure surfaces.

3.2 Finite difference form of the equation

Using the box method eq (10) is approximated as follows :

$$nF_T = 2k_0^2 a^2 \Delta\lambda_i^2 \cos^2\varphi_j \left(\sum_E - \sum_W\right)\left[w_x \overline{P_*}^{\lambda} \frac{T_{ik}(x) - T_{ik}(x)}{a\,\Delta\lambda_i \cos\varphi_j} \cdot \right.$$

$$\left. \sqrt{\left(\frac{1}{a\cos\varphi_j \Delta\lambda}\delta_\lambda u_k - \frac{tg\varphi_0}{a}\overline{v_k}^{\lambda}\right)^2 + \left(\frac{1}{a\cos\varphi_j\Delta\lambda}\delta_\lambda v_k + \frac{tg\varphi_0}{a}\overline{u_k}^{\lambda}\right)^2}\right.$$

$$+ 2k_0^2 a^2 \Delta\varphi_j^2 \left(\sum_N - \sum_S\right)\left[w_x \overline{P_*}^{\varphi} \frac{T_{ik}(x) - T_{ik}(x)}{a\,\Delta\varphi_j} \cdot \right. \qquad (11)$$

$$\left. \sqrt{\frac{\cos^2(\varphi_j \pm \frac{1}{2}\Delta\varphi_j)}{a^2\Delta\varphi_j^2}\left\{\delta_{a\varphi}^2\left(\frac{u_k}{\cos\varphi_j}\right) + \delta_{a\varphi}^2\left(\frac{v_k}{\cos\varphi_j}\right)\right\}}\right]$$

.../12

3.3 Implementation

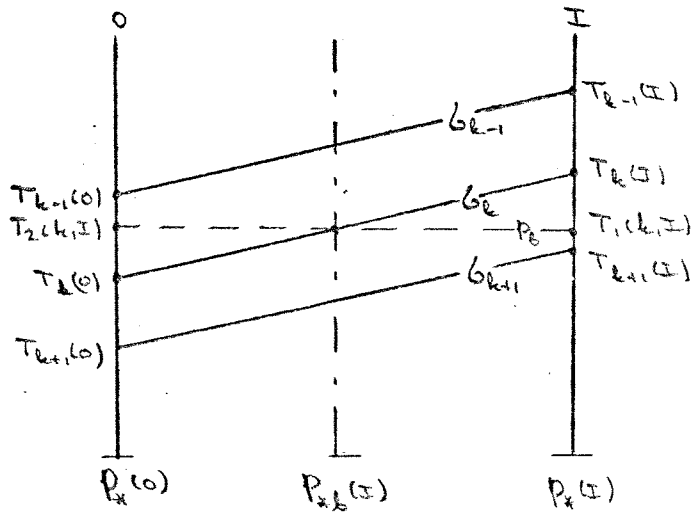
The horizontal T-gradient on a pressure surface is determined, in general, as follows :

1° Calculate the pressure p_b at the main σ-level for each interface between two boxes

2° Determine the σ value of this pressure surface at the centres of the central and neighbour boxes

3° Interpolate linearly between the σ-surfaces to obtain the temperatures on this pressure surface at the central (T_2) and the neighbour (T_1) boxes

Note: In all figures full drawn skew lines are σ-surfaces, dashed horizontal lines are pressure surfaces



.../13

Ad 1° The pressure at the interface between central box 0 and neighbour box I is calculated as follows :

Ground pressure at interface $p^*_b = \frac{p^*(0) + p^*(I)}{2}$

The pressure at the interface on the main σ-level is therefore:

$$p_b = \sigma_k \frac{p^*(0) + p^*(I)}{2}$$

Ad 2° The σ-values $\sigma_p(J)$ at this pressure level in the centres of boxes J are

$$\sigma_p(J) = \sigma_k \cdot \frac{p^*(0) + p^*(I)}{2 p^*(J)} \quad \text{with } J = 0 \text{ en } I$$

Ad 3° For the linear interpolation we want to know if these values $\sigma_p(J)$ are within the range $\sigma_{k+1} - \sigma_{k-1}$, or not. This is done for the upper eight layers in the 400-loop as follows :

We want $\sigma_p(J)$ between σ_{k-1} and σ_{k+1}:

$$\sigma_{k-1} < \sigma_p(J) < \sigma_{k+1}$$

$$\therefore 1 - \frac{\sigma_{k-1}}{\sigma_k} > 1 - \frac{p^*(0) + p^*(I)}{2 p^*(J)} > 1 - \frac{\sigma_{k+1}}{\sigma_k}$$

With $QS(I) = \frac{p^*(0) + p^*(J)}{2 p^*(I)}$ and $QS(I+10) = \frac{p^*(0) + p^*(I)}{2 p^*(0)}$

this inequality reads for centre points (J=0) :

$$1 - \frac{\sigma_{k-1}}{\sigma_k} > 1 - QS(I+10) > 1 - \frac{\sigma_{k+1}}{\sigma_k}$$

and for neighbour points J = I :

$$1 - \frac{\sigma_{k-1}}{\sigma_k} > 1 - QS(I) > 1 - \frac{\sigma_{k+1}}{\sigma_k}$$

.../14

Now the test if 1-QS (I) is between the limits indicated in the previous two equations is not actually performed, but instead an approximated test is done. In the 400-loop it is investigated if |1-QS(I) | is smaller or larger than a certain quantity TOL=.04265 for all levels down to k = 8 (so except for the lowest level one TOL-value has been assumed). It is the same number used in STDPHI. TOL is about equal to

$$\frac{\sigma_{8\frac{1}{2}} - \sigma_8}{\sigma_8} = .042962963$$

and might well be a slide rule accuracy approximation to this number.

In the 400-loop integer array elements NQS (I) and NQS (I+10) are defined as follows :

Central point :

NQS (I+10) = 0 if $\sigma_p < \sigma_{k-1}$ or $\sigma_p > \sigma_{k+1}$

= -1 if σ_p inside this range and p* (I) $\leq$ p* (o)

= +1 if σ_p inside this range and p*(I) > p* (o)

neighbour points:

NQS (I) = 0 if $\sigma_p < \sigma_{k-1}$ or $\sigma_p > \sigma_{k+1}$

=-1 if σ_p inside this range and p*(o) p* $\leq$(I)

=+1 if σ_p inside this range and p*(o) p* >(I)

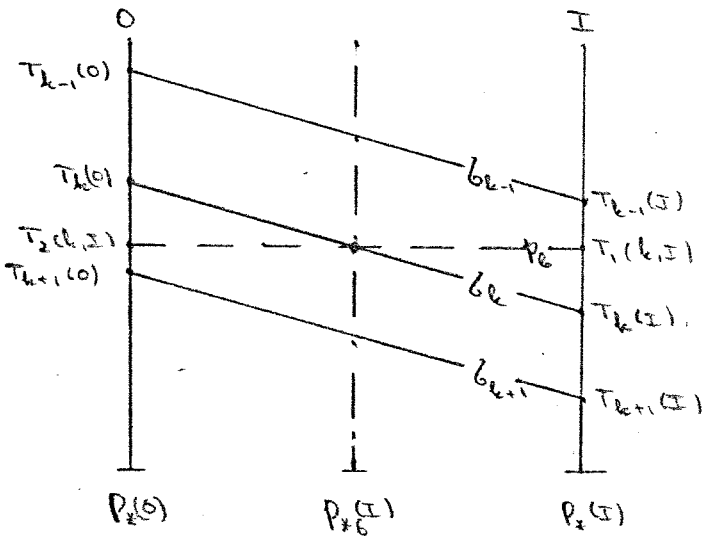
Next the subroutines TR1 and TR2 are called. We consider only TR1; TR2 is analogous.

.../15

TR1

In this subroutine the three NQS-values are treated separately.

1^{o} NQS = -1



T_1 (k,I) is obtained by simply linear interpolation between T_{k-1} and T_k :

$$\frac{T_1(k,I) - T_{k-1}}{T_k - T_{k-1}} = \frac{\sigma_p - \sigma_{k-1}}{\sigma_k - \sigma_{k-1}}$$

which implies (k = 2,9)

$$T_1(k,I) = \left[\left\{\sigma(k) - \sigma(k)\cdot\frac{p_{*-}(O)+p_{*-}(I)}{2p_{*-}(I)}\right\}\cdot T_{k-1-}(I)\right.$$

$$\left. + \left\{\sigma(k)\frac{p_{*-}(O)+p_{*-}(I)}{2p_{*-}(I)} - \sigma(k-1)\right\} T_{k-}(I)\right] \Big/ \left(\sigma(k) - \sigma(k-1)\right)$$

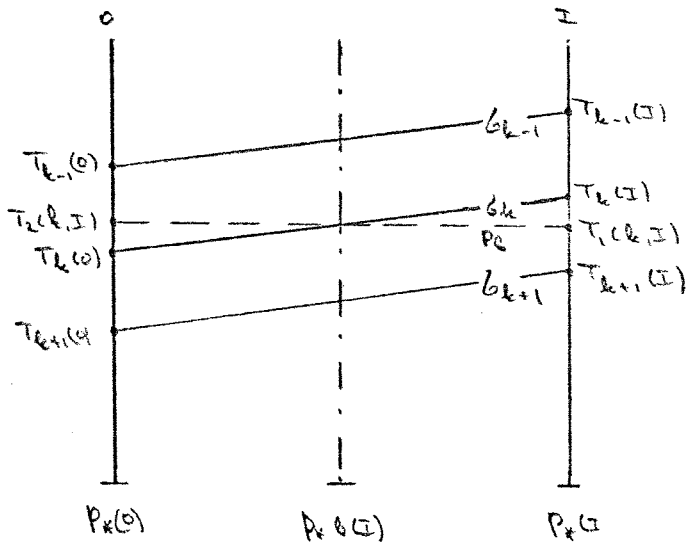
If however σ_k is the highest σ-level (k=1) then T_1 (k,I) is obtained by extrapolation : $\frac{T_1(k,I) - T_{k-1}}{T_k - T_{k+1}} = \frac{\sigma_p - \sigma_{k+1}}{\sigma_k - \sigma_{k+1}}$

../16

implying

$$T_1(1,I) = \left[\left\{\sigma(2)-\sigma(1)\cdot\frac{p_{*-}(0)+p_{*-}(I)}{2p_{*-}(I)}\right\}T_{1-}(I) + \left\{\sigma(1)\frac{p_{*-}(0)+p_{*-}(I)}{2p_{*-}(I)} - \sigma(1)\right\}T_{2-}(I)\right] / (\sigma(2)-\sigma(1))$$

2^{o} NQS = +1



Again $T_1(k,I)$ is obtained by linear interpolation, in this case between T_k and T_{k+1}. If σ_k is the lowest level, extrapolation might produce unrealistic high temperatures, therefore in this case $T_1(k,I)$ is approximated by averaging the temperatures of central and neighbour point at the main σ-level :

$$T_1(k,I) = \tfrac{1}{2}\left[T_{9-}(I) + T_9(0)\right]$$

which corresponds to the supposition that the temperature is constant on the pressure surface p_b .

.../17

3° NQS = 0 : call TSTT

In TSTT at first a quantity QSK is defined :

$$QSK = \sigma(k)\ \frac{p*_{-}(0) + p*_{-}(I)}{2p*_{-}(I)} = \sigma_p(k)$$

Three cases are distinguished :

a) $\sigma(2) < \sigma_p(k) \leq \sigma(9)$

In the 10-loop a search is made for the level in such that $\sigma(m-1) \leq \sigma_p(k) \leq \sigma(m)$ and next $T_1(k,I)$ is determined by linear interpolation between $\sigma(m)$ and $\sigma(m-1)$

$$T_1(k,I) = T\{\sigma(m) - \sigma(k) \cdot \frac{p*_{-}(0) + p*_{-}(I)}{2p*_{-}(I)}\} T_{m-1}(k) +$$

$$+ \{\sigma(k) \frac{p*_{-}(0) + p*_{-}(I)}{2p*_{-}(I)} - \sigma(m-1)\} T_2(k)] / (\sigma(2) - \sigma(1))$$

b) $\sigma_p(k) \leq \sigma(2)$

In this case m is set equal to 2 and the same interpolation formula as for a) is used. This amounts to interpolation in case $\sigma(1) \leq \sigma_p(k) \leq \sigma(2)$ and extrapolation if $\sigma_p(k) \leq \sigma(1)$.

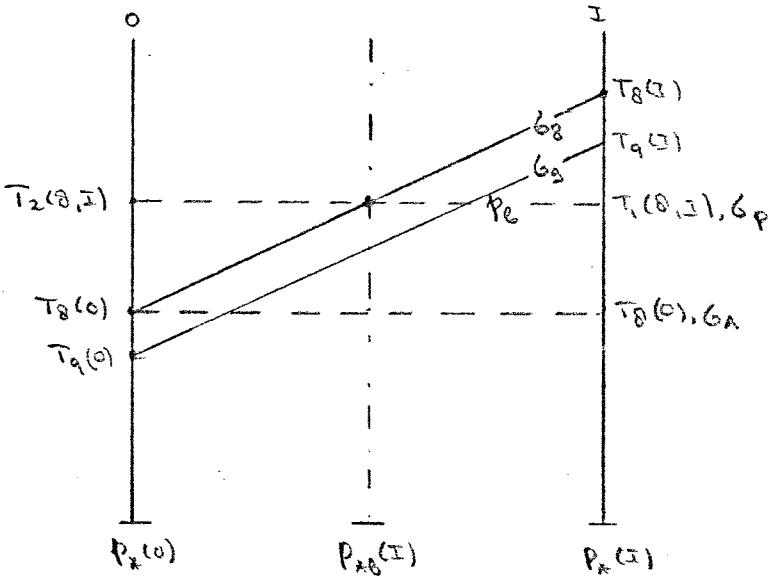
c) $\sigma_p(k) > \sigma(9)$

Again two cases are distinguished :
I) If the main σ-level to which p belongs is also 9 (i.e. k=m=9) then it is assumed that the temperature is constant on pressure level p_b so :

$$T_1(I,k) = \tfrac{1}{2}\,[T_{9-}(0) + T_{9-}(I)]$$

.../18

II) If the main σ-level, to which p belongs is < 9 ($k < m=9$) then the following picture applies (with, for example, k=8)



At the neighbour point the σ-value σ_A is determined by the pressure level, corresponding with σ_8 in the central point. Next it is suppos that the temperature at this level is constant and equal to $T_8(0)$. Finally, T_1 (8) is calculated by linear interpolation between σ_A and σ_9:

$$T_1(8,I) = \frac{(\sigma_A - \sigma_p) T_9(I) + (\sigma_p - \sigma_9) T_8(0)}{\sigma_A - \sigma_9}$$

with $\sigma_p = \sigma_k \frac{p_{*-}(0) + p_{*-}(I)}{2 p_{*-}(I)}$ and $\sigma_A = \sigma_k \frac{p_{*-}(0)}{p_{*-}(I)}$

With the calculated T_1 and T_2, finally, HFT is calculated in HORZDF.

.../19

Chapter 3 - The I/O scheme

The I/O scheme for the GFDL model is designed to work for regular lat-long and for modified Kuri grids. The data is kept on a (direct access) disc and is stored sequentially, the data for the first point in the first row occurring first and the data for the last point on the last row occurring last. Each point has a grid ##** beginning at 1,2 ..... NOPTS where NOPTS is the ##** of points on the grid. The data is organised on the disc in blocks of length BUFSIZ, except possibly for the very last block which can be of length less than BUFSIZ.

In core we have a circular buffer called PTBUFF where we have space for

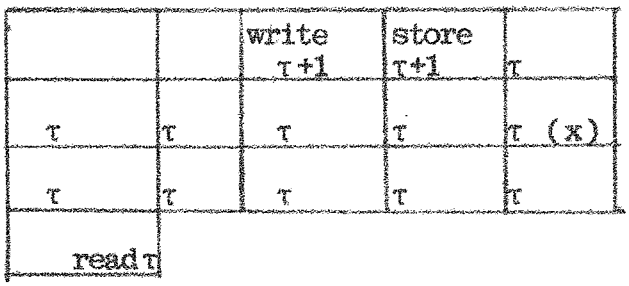
1 read block where reading from disc is being done for τ level data

1 write block where writing to disc is being done for $\tau+1$ level data

1 store block where results at time level $\tau+1$ are being stored before being written out.

N blocks where N is large enough to accommodate data for all the points between the N neighbour on an LL* grid (NW on a MK, modified Kuri) and the S neighbour (on LL) or SE neighbour (on MK), and all the intervening points.

To fix ideas suppose we have an LL grid with S blocks per row. Then the maximum data requirement is illustrated on the sketch if we are computing in the block marked (x).



* LL stands for Latitude Longitude grid
MK " " Modified Kurihara grid

** The sign ## designates "NUMBER" throughout this chapter.

.../20

If we have M blocks per latitude then PTBUFF should be at least (2*M+4) blocks long. The scheme is designed to handle blocks of arbitrary length.

As soon as we use the first word of the last block of τ level data we initiate a write for the next block.

As soon as we have stored the $\tau+1$ level data for the first point of the store block we initiate a write for the previous block.

The computation is not done on the buffer PTBUFF but rather on a small buffer called CLASS where we have τ level data for the current key point and all its neighbours together with space for the $\tau+1$ results at the key point.

To begin the computation for a new point we call a routine called NEXT which does the following

1. Moves out from CLASS the $\tau+1$ data for the previous point to the store block of PTBUFF
2. Moves in from PTBUFF to CLASS the τ level data at the current point and its neighbours.

In order to do these moves NEXT must have pointers to the point in PTBUFF where the data is to be moved to/from.

These pointers are found by successive calls to the function subroutine NEXTPT whose argument is $\pm$ the grid ## of the point in question. If the argument is negative NEXTPT returns a pointer to the place where the ($\tau+1$) level data is to be stored. If the argument is positive NEXTPT returns a pointer to the place where the τ level data for the grid point can be found.

.../21

In addition NEXTPT performs two other functions. If the point where the (τ+1) level data is to be stored is the first word of a block then it initiates a write to disc for the previous block. If the point where the τ level data is to be found is the first word of the last block of τ level data in core then it initiates a read for the next block.

At the beginning of a pass through the grid we need to know where in PTBUFF we should start storing, start reading and how many blocks we should read to begin with. On an LL grid the answers are straightforward if the block length is a sub-multiple of the ## points in a row. The answers are not so clear if the block length is not a sub-multiple or if we are on a Kuri grid. This is where the simulation program is used.

The simulation program is essentially the same code as the routines NEXT and NEXTPT. It simulates a read by setting the appropriate block of PTBUFF to plus the grid ## of the data to be read. It simulates a store of (τ+1) data by setting the appropriate area in PTBUFF to minus the grid ## of the point being stored. When it simulates the write it checks that this negative value is as it should be. It also carries two variables INDEX, NNDEX,which are respectively the core numbers in PTBUFF of the first word of the block currently being read or written.

In NEXTPT an error message is given if (i) INDEX = NNDEX indicating that we are reading data into a block from which we are also writing (ii) If, when we are writing out, the contents of PTBUFF do not correspond to the negative of the appropriate grid ##.

In NEXT a check is made,everytime we use τ level data, that the contents of PTBUFF is indeed the grid ## of the correct point. If not then an error message is issued.

.../22

The simulation program requires the following data

NRES = ## of latitudes between equator and pole
NPPR = ## of points per row
BUFSIZ
= ## of points in a block
NOBUF
= ## of blocks in PTBUFF
IOUTA
= point in PTBUFF where first store is to be made
IREADA
= point in PTBUFF where first read is to be made
IBUFIN
= ## of blocks to be read at start of pass.

If BUFSIZ is less than or equal to the ## of points on the first row then we must read enough blocks to achieve the following :

1) Provide data at all the neighbour points for the first point in the grid. This implies that we have all the first latitude and at least the first few points on the second row.

2) Ensure that the first point of the last block read in is not on the first row.

If BUFSIZ is larger than the ## of points on the first latitud then we need only read in 1 buffer to begin with.

F L O W D I A G R A M F O R NEXT IN SIMIO

KROW - the row ## of the next row - initialised in main to 1

NROW - KROW-1

NPOINT- point number in row - initialised in main to 0
Coming into NEXT it is the point number of the previous point.
Then incremented by 1 after τ+1 data for that point has
been sent out to PTBUFF

NPBROW =grid no of last point in previous row. The values are stored
in NPBROE - initialised to zero in main

NPIROW= ## of points in current row - initialised to zero in main.

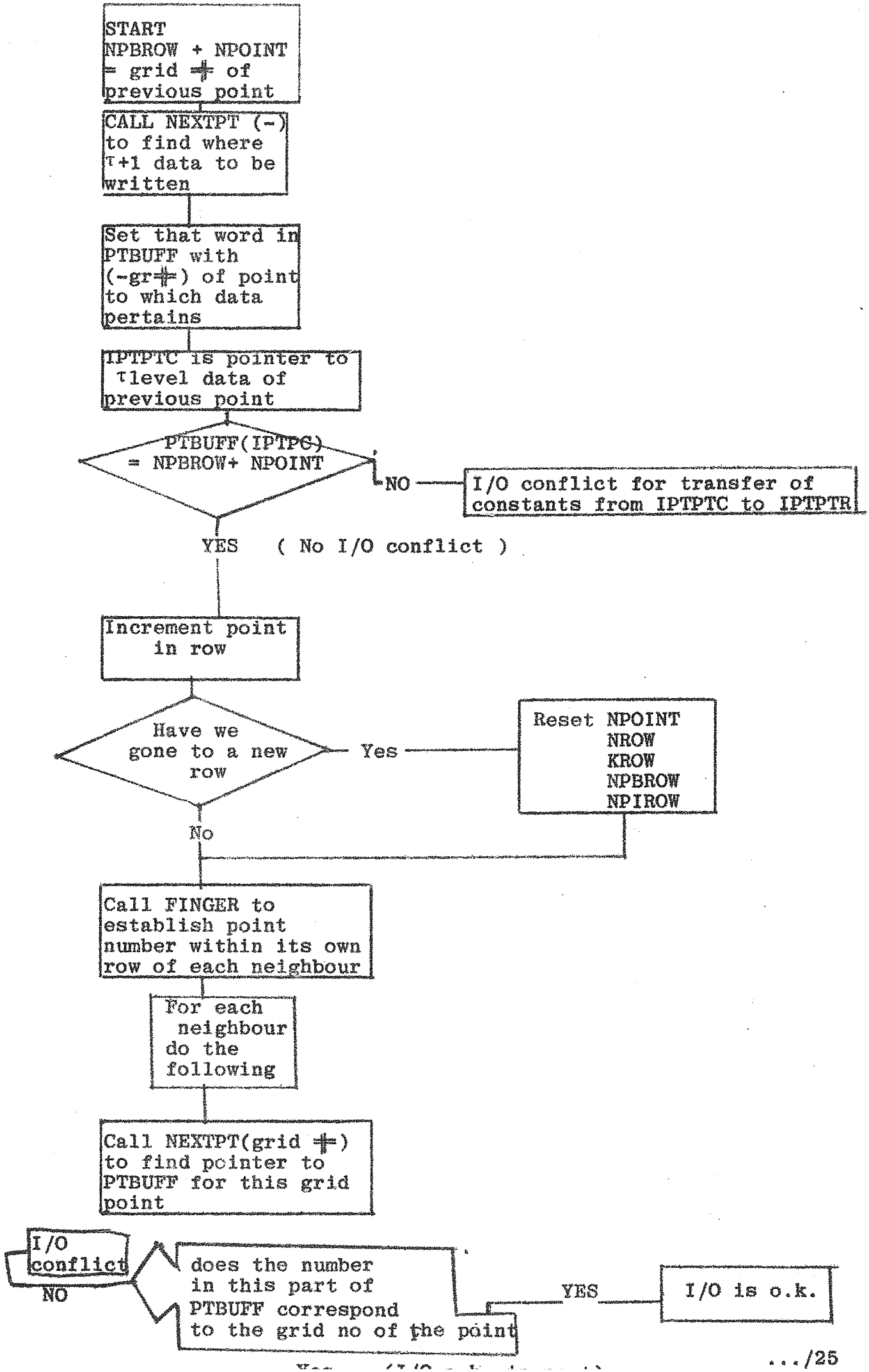
IPTPTC= Coming into main it is the pointer to the place in PTBUFF for
data of previous point. Initialised to 1 by data statement.
After τ +1 data has been sent out to PTBUFF for the previous
point it is redefined by a call to NEXTPT as the pointer
to the τ+1 data for the current central point.

KPTNBR= Array containing point number within its own row of surroundin
points,got from FINGER.

The simulation works as follows :
As the τ level data is read in from disc the area in PTBUFF to which it is read is set to the grid point number of the point to which the data pertains.

As the τ+1 level data is written out from CLASS to PTBUFF the area in PTBUFF to which it is written is set with the negative of the grid point number to which it pertains. As the constants are transferred from one part of PTBUFF to another in the write out part of NEXT, and as the data is read in from PTBUFF to CLASS in the second part a check is made that the contents of the point in PTBUFF from whi the transfer is made is precisely plus the grid point value of the point to which the data pertains. If it is not then the data has been over written and we get an I/O conflict in NEXT.

.../24


START
NPBROW + NPOINT = grid # of previous point
CALL NEXTPT (-) to find where τ+1 data to be written
Set that word in PTBUFF with (-gr#) of point to which data pertains
IPTPTC is pointer to τlevel data of previous point
PTBUFF(IPTPC) = NPBROW+ NPOINT
NO
I/O conflict for transfer of constants from IPTPTC to IPTPTR
YES (No I/O conflict)
Increment point in row
Have we gone to a new row
Yes
Reset NPOINT NROW KROW NPBROW NPIROW
No
Call FINGER to establish point number within its own row of each neighbour
For each neighbour do the following
Call NEXTPT(grid #) to find pointer to PTBUFF for this grid point
I/O conflict
NO
does the number in this part of PTBUFF correspond to the grid no of the point
YES
I/O is o.k.

.../25

FLOW DIAGRAM FOR NEXTPT IN SIMIO

NEXT PT — is given as argument a grid point ≠ and returns a pointer which tells where in PTBUFF the τ level data for that point is kept (positive argument) or where in PTBUFF the $\tau+1$ level data for that point is to be written (negative argument). It is called with zero argument twice, at the beginning and end of a pass. At this time it returns the value 1 at beginning of pass and LNDEX at end of pass.

In addition it performs the I/O to disk as required.

Variables

IREADA	position in PTBUFF to which first read is made
IOUTA	" " " from which first write is made
IBUFIN	≠ of buffers read in to begin with. Must be sufficient to provide data for first block (i.e. on latitude and at least one point to south of 1st point).
BUFSIZ	≠ of points in one buffer
NOBUFF	≠ of buffers in PTBUFF
TBUF	BUFSIZ * NOBUF
INPTS	grid ≠ of first point of next block to be read in
INPTSP	grid ≠ of last point of next block to be read in
INDEX	core ≠ of 1st word of next block to be read in unl we are at the beginning of a pass when it points to core ≠ of 1st word of last block that has been read.
INDEXP	core ≠ of last word of next block to be read in
LASTPT	grid ≠ of 1st word of second last block read in
NNDEX	core ≠ of 1st word of next block to be written out initialised to -1 at start of pass
OUTPTS	grid ≠ of first point of next block to be written out
OUTPTSP	grid ≠ of last point of next block to be written ou
NNDEXP	core ≠ of last word of next block to be written out
BLKNO	block ≠ of next block to be read or written. Used to route I/O to correct section of mass storage
BUFSIZ	≠ of points in 1 buffer
LASTPT	grid ≠ of 1st point in second last buffer that has been read in

If BUFSIZ is less than or equal to ≠ of points in first row then at the beginning of a pass we must read in enough blocks to achieve the following

1) provide data at all the neighbour points for the first point on the grid. This implies that we have all the first latitude (to get E,W neighbours) and the first o or two points on the next row in core.

.../26

2) In addition we must ensure that the first point of the last block read in is not on the first row . (These two considerations determine the size of IBUFIN = NPPR (1) / BUFSIZ +2).
Then LASTPT is set to the grid ╪ of the first word of the next to last block.
Thus we have the block beginning with (LASTPT + BUFSIZ in core . When we first use the data for the point (LASTPT + BUFSIZ) we initiate a read for the next bloc (the first word of which has grid ╪ (LASTPT + 2*BUFSI and increment LASTPT by BUFSIZ.

If BUFSIZ is greater than the ╪ of points on the first row, the start up procedure is a little different.
We set IBUFIN =1.
At the beginning of a pass NEXT calls NEXTPT with argument 0. One block is read in and LASTPT is set to (-BUFSIZ + 1)< 0
NEXT immediately calls NEXTPT with argument 1.
Since 1 (=LASTPT + BUFSIZ) is the 1st word of the l block in core a read for the block beginning at 1+ BUFSIZ (LASTPT + 2 BUFSIZ) is initiated and LASTPT is reset to 1.

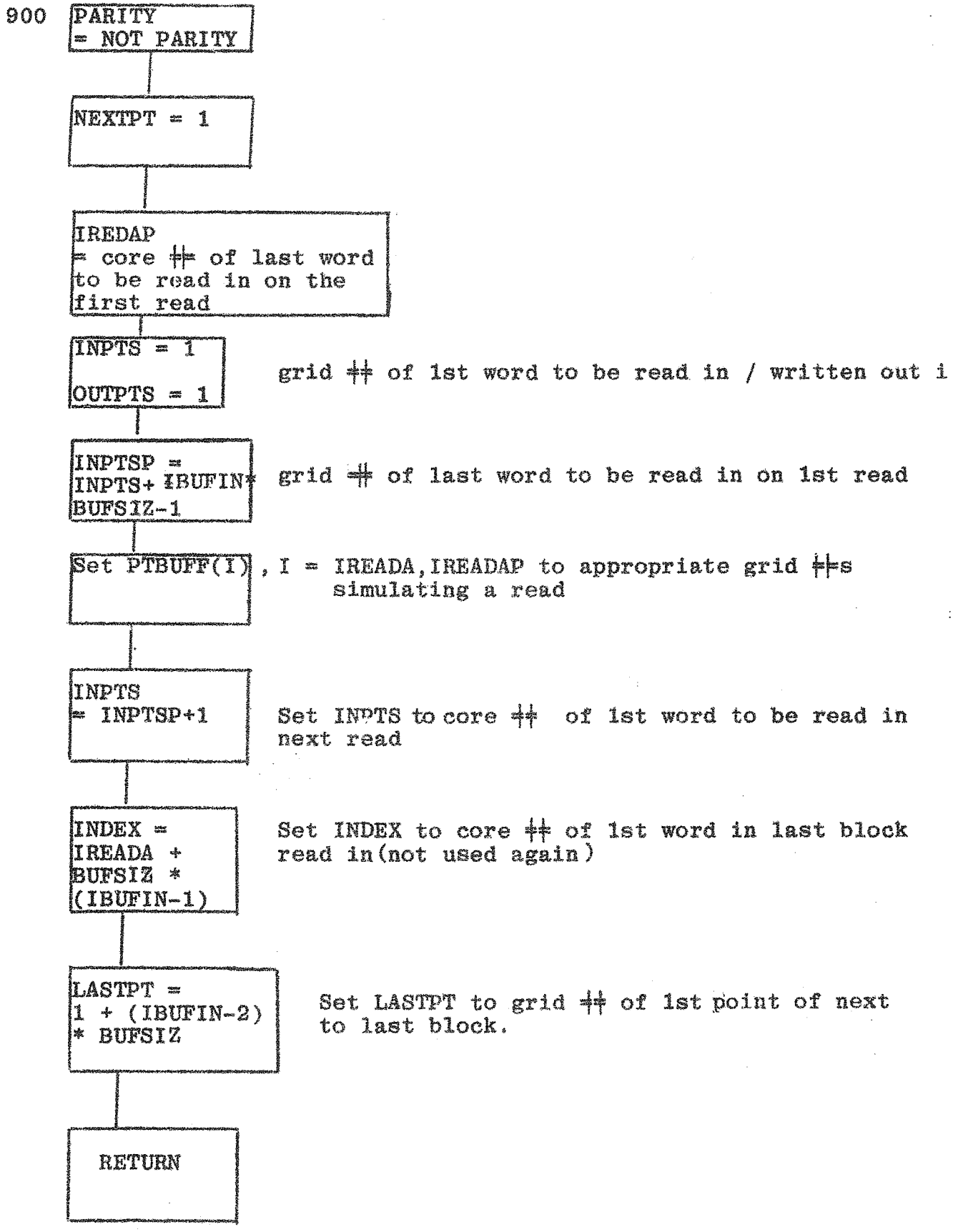
Thereafter the procedure is as above. The first time we access the data for LASTPT + BUFSIZ a read i the block beginning at LASTPT + 2*BUFSIZ is initiated.
Whenever we initiate a read for a block INDEX is set to the core ╪ of the first word in that block.

We will flow diagram the four parts of NEXTPT separately:

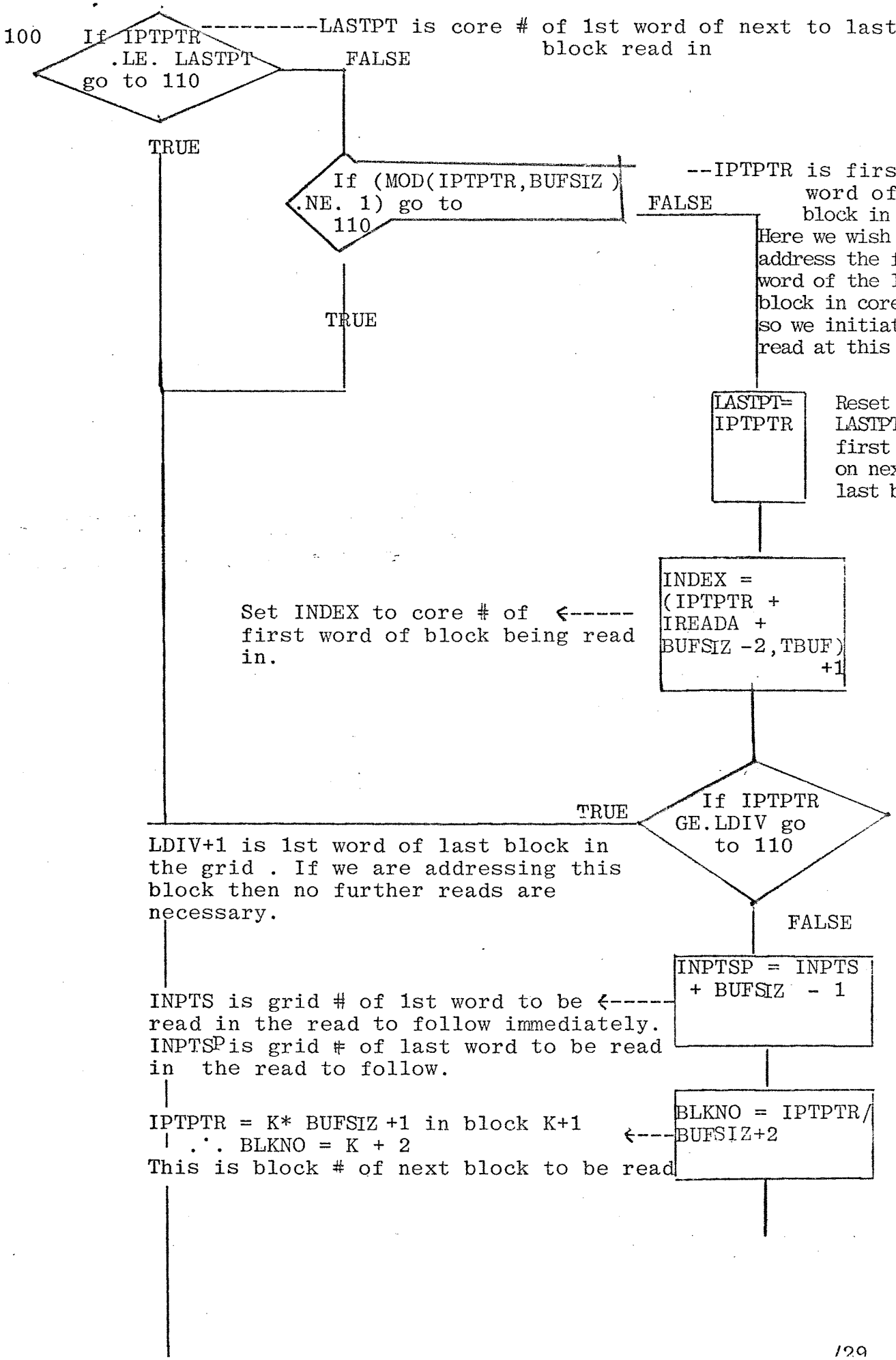
1) Beginning of pass IPTPTR = 0, PARITY = TRUE initially
2) Positive argument reading in from disc if necessary
3) Negative argument writing out to disc if necessary
4) End of pass IPTPTR = 0, PARITY = False initially.

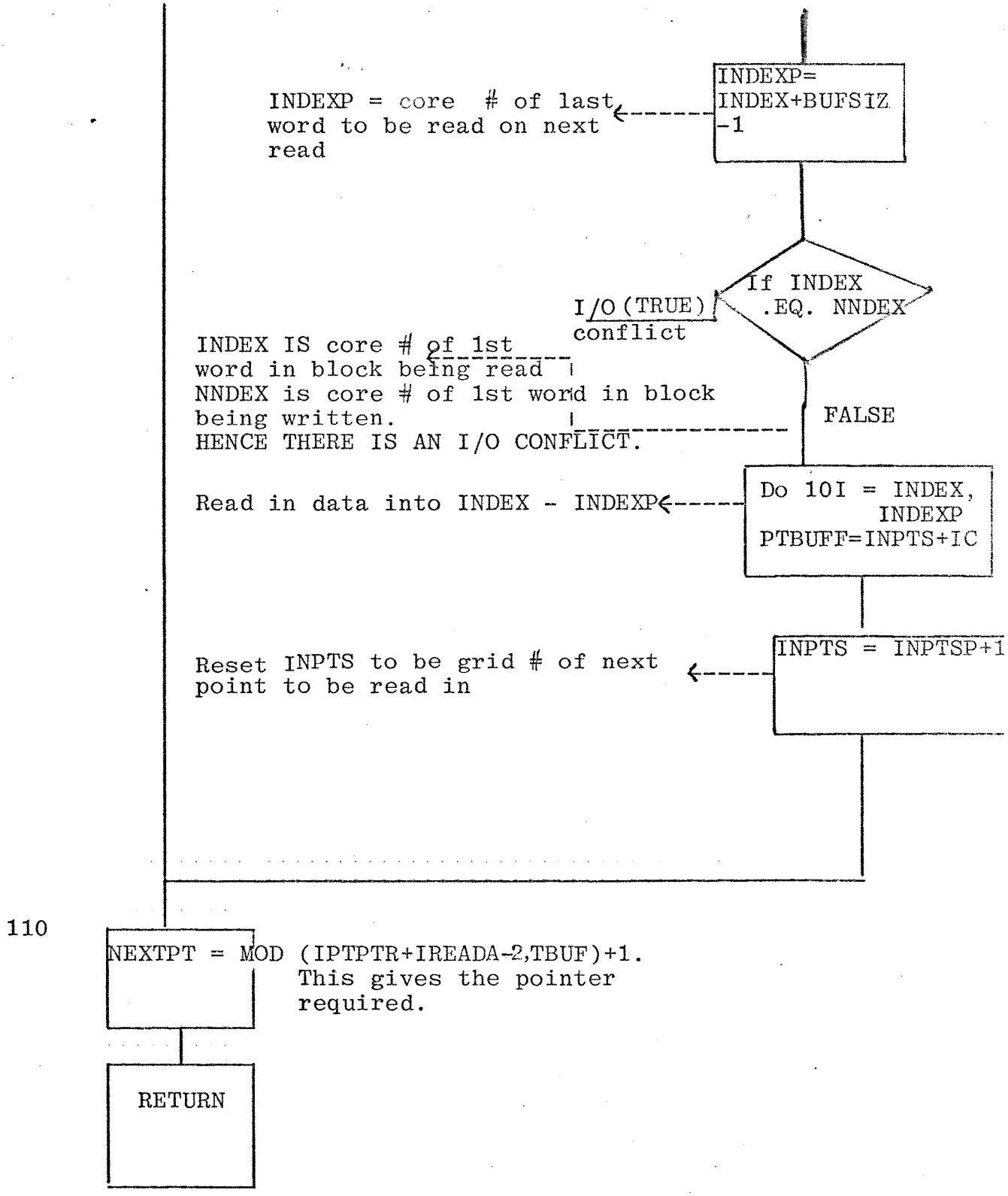
.../27

1) PARITY = TRUE, start of pass

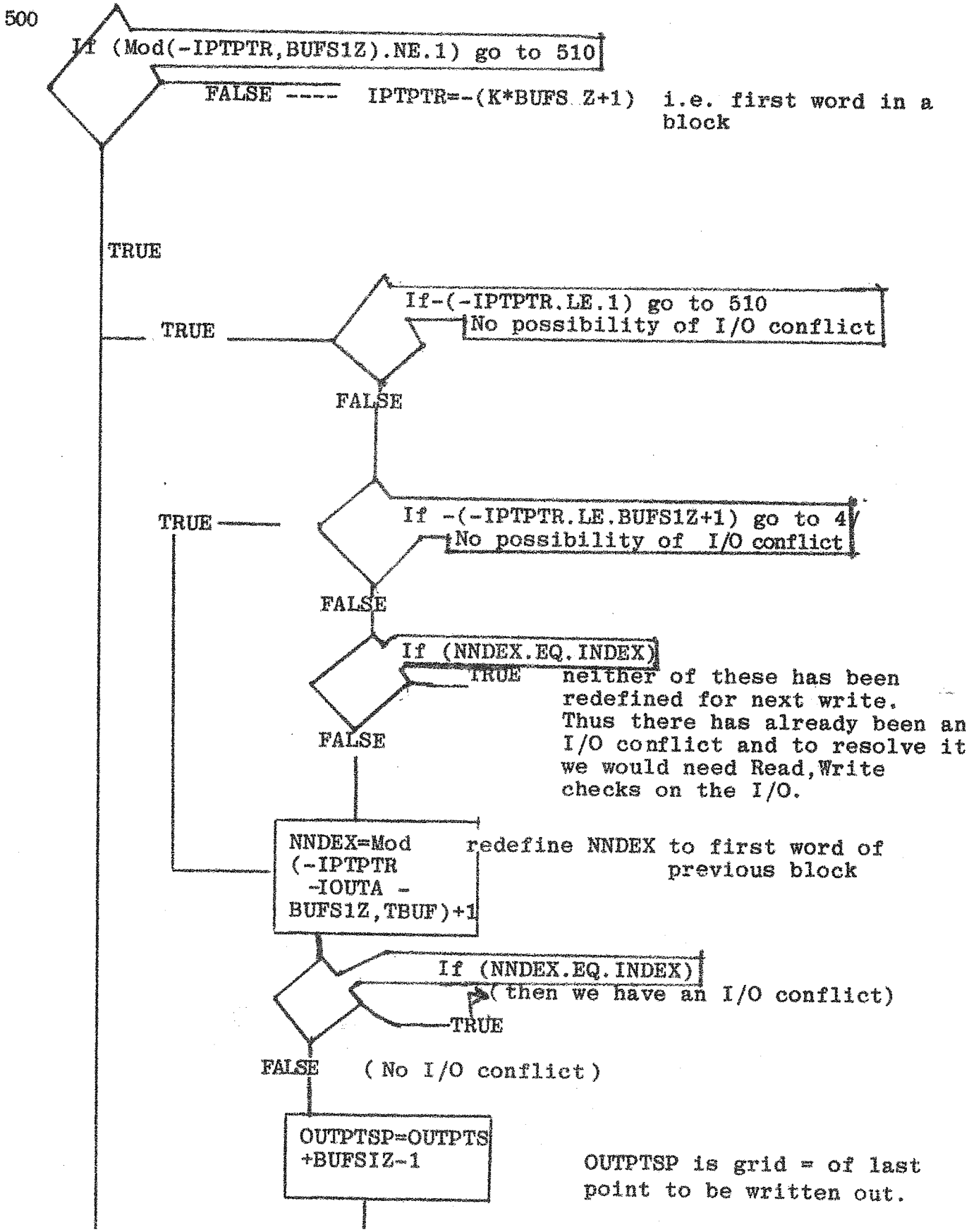


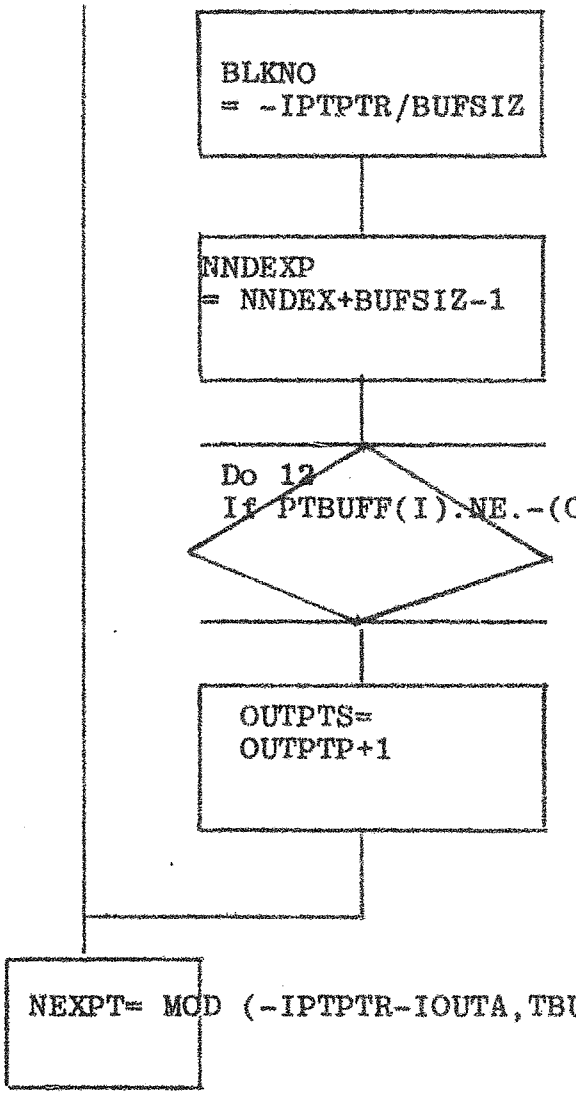
2) Positive argument. Return core # of level data for IPTPTR. Read in if necessary.




INDEXP= INDEX+BUFSIZ -1
INDEXP = core # of last word to be read on next read
If INDEX .EQ. NNDEX
I/O (TRUE) conflict
INDEX IS core # of 1st word in block being read
NNDEX is core # of 1st word in block being written.
HENCE THERE IS AN I/O CONFLICT.
FALSE
Do 10I = INDEX, INDEXP
PTBUFF=INPTS+IC
Read in data into INDEX - INDEXP
INPTS = INPTSP+1
Reset INPTS to be grid # of next point to be read in
110
NEXTPT = MOD (IPTPTR+IREADA-2,TBUF)+1.
This gives the pointer required.
RETURN

3) Negative argument. Return core # of place where τ+1 data for IPTPTR is to be stored.
NNDEX = -1 at start of pass





This sets the block number for mass storage of the block to be written out.

This sets NNDEXP as last word of block to be written out

sets pointer for next store.

.../32

4)End of a Pass

At the end of a pass there are 3 calls to NEXTPT

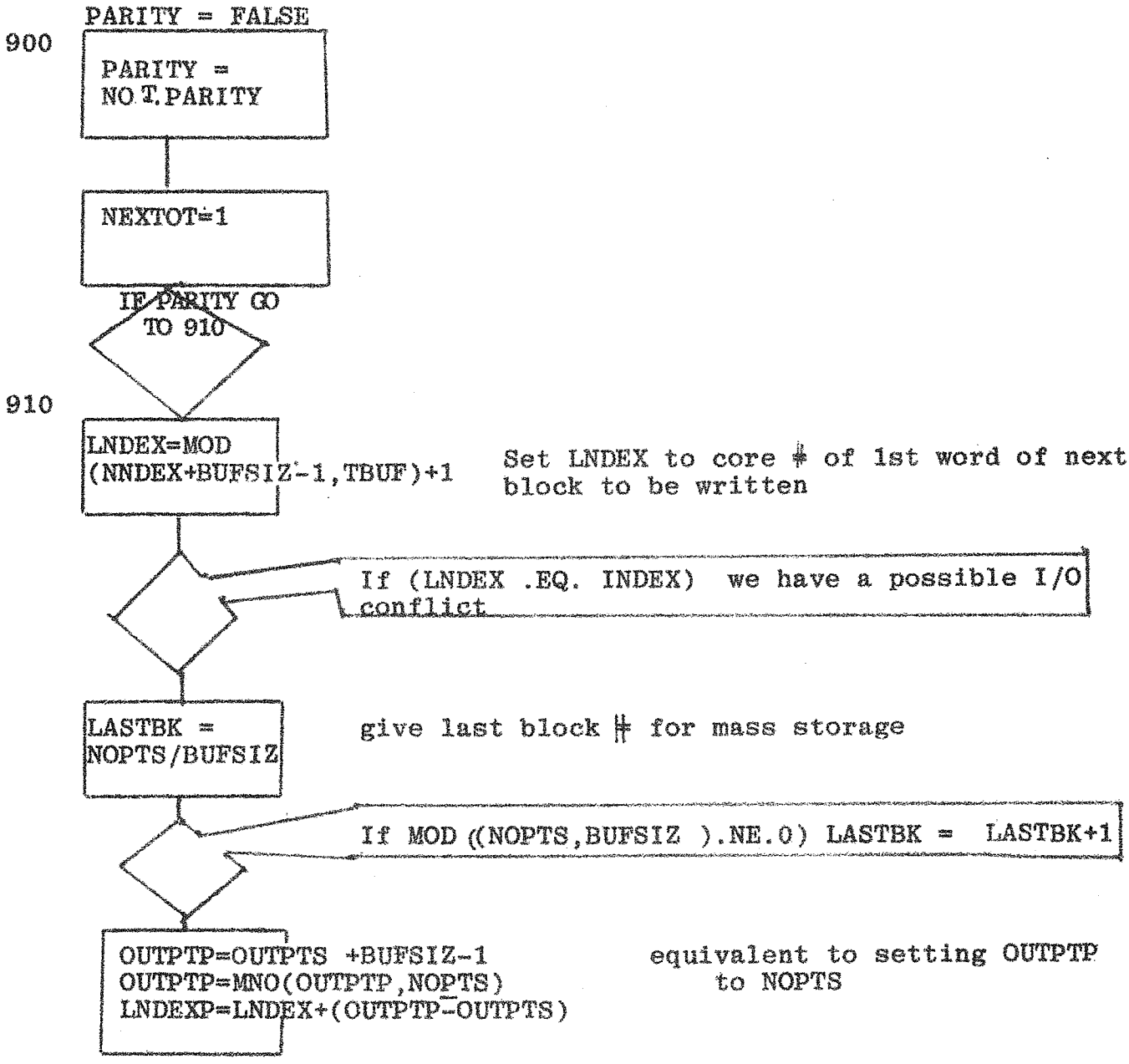
1) NEXTPT (-NOPTS) NOPTS = # of points
   This tells where to store the τ +1 values for the last point

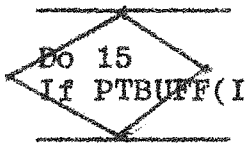
2) NEXTPT (NOPTS)
   This tells where to find the KPTNBR values to be stored with the τ +1 data

3) NEXTPT (0)
   This does the final write out.

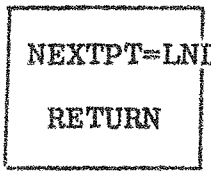
In the full model these are done in a separate routine ENDTIM

Here we discuss the lcst call to NEXTPT (0). At the beginning of this we have parity = false





go to 13 - I/O conflict



set NEXTPT to this to check for I/O conflict at beginning of next pass.

.../34

Chapter 4 - Namelist Input parameters

In subroutine CONSOL a number of input parameters can be read in, using the (non standard) FORTRAN facility NAMELIST. This NAMELIST is called INPUT. Most of these input parameters have default values, defined in BLOCK DATA. Therefore only input parameters,which have no default value or should have a value different from the default value, are to be specified in the NAMELIST. For the precise format of NAMELIST we refer to the manual of the relevant computer. Table 1 summarises all relevant NAMELIST input parameters, their default values and their meaning.

Notes:

1. On the CDC 6600 it is impossible to read in character strings via NAMELIST. The code on the CDC 6600 has been changed such that numbers can be read in for the parameters TYPE and TYPESM with the following meaning :

'TAPE'	1
'DISK'	2
'SPEC'	3
'WAVG'	4

2. N24 integrations on the basis of the available 1 March 1965 data set, using spectral filtering near the pole (TYPESM='SPEC') are unstable and blow up after some hours model time. Therefore weighted averaging (TYPESM='WAVG') should be used.

TABLE 1 : NAMELIST INPUT PARAMETERS

Input Parameter	Default	Comments
MRSTR	none	Start up KTAU; equal to last step completed needed only if TYPE='TAPE'.
KTAUST	99999	Stop KTAU.
TYPESM 1)2)	'SPEC'	Type of smoothing at poles. 'WAVG' = weighted averaging, 'SPEC' = spectral filtering.
CRASH	.FALSE.	If .TRUE., picking up after crash or something else such that integrals have not been copied from disk to tape.
NEWINT	.FALSE.	If .TRUE., no integrals on Integral Tape data set.
TYPE 1)	'DISK'	Starting up from data on 'DISK' or 'TAPE'.
VBEG	.FALSE.	If .TRUE., preformat units 14 and 15.
LABEL	the experiment label	Experiment label for CTLBLK
IHIS	21600	Interval in seconds between history tape writes
ISMTH2	N24:120 N48:216	Do Euler Backward every ISMTH2 steps, with a displacement of IDISP2, for NSMTH2 steps.
NSMTH2	N24:-1 N48:215	
IDISP2	1	
ISMT	53	Do time smoothing every ISMT steps.

1) see note 1. on previous page

2) see note 2. on previous page

.../36

Chapter 5 - User instructions

1. The N24 version on the Met.Office IBM 360/195.

1.1 Source Code

1.1.1 Data set DSN=ECMWF.SRCE.METO43 contains the source text of the program as a set of members, each containing one or a few subroutines. Table 2 gives a summary of the member names and the corresponding subroutines.

1.2 Object Code

1.2.1 Data set DSN=ECMWF.OBJ.METO43 contains the object code of the following necessary subroutines:
ESATUR (an object code version equivalent to the above mentioned Fortran version)
FORTBDAM (a set of subroutines for direct access I/O)
CPUT (an IBM CPU timing subroutine)

1.2.2 Data set DSN=M02.OBJLIB contains the time-of-the-day subroutines
BJZTIME
BJTODAY

1.2.3 Data set DSN=M22.OBJLIB contains a subroutine, which writes time of day and date in an appropriate format:
DAT2CF

1.2.4 Data set DSN=MET.PROGLIB contains a date subroutine
ZPDATE

.../37

1.3 Load module

1.3.1 Data set DSN=ECMWF.LOAD.MET043 contains a load module, including all above mentioned subroutines except the following redundant ones : ANTL,CDAT,COMP, IPMA,SOLA,ZENI, ESATUR (Fortran version).
This load module is created, using FORTH,OPT=2 on the IBM 195. Moreover, in order to simplify restart (see below) lines 00014300-00014900 of subroutine CONST have been deleted.

The member name of this load module is TEMPNAME

1.4 Location of data sets

The above mentioned data sets can be found on the following disks :

DSN=	UNIT=	VOL=SER=
ECMWF.SRCE.MET043	3330	MET043
ECMWF.OBJ.MET043	3330	MET043
M02.OBJLIB	3330	SYS005
M22.OBJLIB	3330	SYS006
MET.PROGLIB	3330	SYS002

1.5 Work disk files

On disk MET043 six disk files have been prepared corresponding with the work files required by the program. Table 3 summarises the essential data.

1.6 Input tape

The input data set of 1 March 1965 are on a magnetic tape which can be attached with the following set of JCL cards

```
//GO.FT12F001 DD UNIT=TAPE9,DISP=OLD,VOL=SER=HOLY2,
//    LABEL=(1,NL,,IN),
//    DCB=(DEN=3,RECFM=VS,BLKSIZE=11824)
```

.../38

1.7 Restart

For long runs (> 10 min. CPU time) Met. Office requires appropriate restart facilities. This means that restart from the work disk files should be possible by the operator. Apart from minor changes in the name list input parameters (see Chapter 4) at the very first restart, the main problem is that it is necessary for the system to know how many history files have been written to tape. For this purpose subroutine CONSOL has to be modified. The modified CONSOL starts with calculating the number of written history files. Next all history files on file 10 are skipped by READ statements, using an END=parameter, the execution of which causes an increase of the file counter with 1. At the end of this procedure the file counter has the right value for the next history file.
Fortran changes of CONSOL:

```
      NHIST=KTAU/60                                00014210
C     THE ABOVE STATEMENT IS OF COURSE             00014211
C     DEPENDENT ON FREQUENCY OF HIST.FILE          00014212
C     WRITING                                      00014213
      REWIND 10                                    00014220
      DO 100 J=1,NHIST                             00014230
      IF (NHIST.eq.Ø) GOTO100                      00014240
      DO 101 K=1,114                               00014250
C     114 IS NUMBER OF RECORDS ON                  00014251
C     N24 HISTORY FILES                            00014252
      READ (10,END=100)                            00014260
 101  CONTINUE                                     00014270
 100  CONTINUE                                     00014280
```

These changes have not been incorporated in load module TEMPNAME, and should be included using the source code of CONSOL, changed with the help of METMERGE (see example below).
To further simplify restart, lines 14300-14900 in CONST have been deleted.

.../39

Because the STOP facility has not been implemented in the N24 version, the operator can only stop a run by an operator DROP. The operator should be instructed to drop the job only when the tape drives are not active. Otherwise a history file might be lost. This method, although admittedly very sloppy, gives no problems in practice. Otherwise the STOP facility can easily be implemented. For details see the next paragraph on the N48 version.

1.8 Example

Table 4 reproduces the JCL for a 24 hour run. Six hours have been completed already and this run starts from the work files. Every 6 hours (60 time steps) a history file is written directly to magnetic tape. The job is made fully restartable.

After the jobcard, subroutine CONSOL is changed, using METMERGE, to make the job restartable (see 1.7 Restart). The modified CONSOL is compiled and in the LKED stage linked with load module TEMPNAME. Next 8 history file data sets (FT10F00n) are defined on magnetic tape 410015, followed by the other data sets (see table 3). No data set definition of the input tape FT12F001 is required because this job restarts from work files and not from input tape.
FT13F001 is a necessary scratch disk file which does not need to be catalogued.
FT29F001 is a special output file on which a summary of important job data and results is written. In this case this file is directed to the line printer.

Finally the JCL ends with NAMELIST INPUT (see Chapter 4).

```
//FMHSAE01 JOB (ECMWAE,D),'BAEDE',PRTY=1,REGION=472K,TIME=110
//*INFORM SYSTEM=P95,SETUP=YES,CPU=HIGH
// EXEC METHGE,DSN=ECMWF.SRCE,MET=43,UNIT=3330,
//  VOL=SER=MET043,PRTY=10
//HRGE.SYSIN DD *
./ CHANGE NAME=CONS,NEW=PS
      NHIST=NTAU+0
      REWIND 10                                                          0001422
      DO 100 I=1,NHIST                                                   0001423
      IF(NHIST.EQ.0)GO TO 100                                            0001424
      DO 101 K=1,11                                                      0001425
      READ(10,END=100)                                                   0001426
  101 CONTINUE                                                           0001427
  100 CONTINUE                                                           0001428
./ DELETE SEQ1=00014300,SEQ2=00014400
// EXEC FORTHCLG,OPRTY=10,PARM.FORT="NOSOURCE,OPT=2,ID",TIME.GO=110
//FORT.SYSIN DD DSN=&TEMP,DISP=SHR
//LKED.LOAD DD DSN=ECMWF.LOAD.MET043,DISP=SHR,UNIT=3330,VOL=SER=MET043
//LKED.SYSIN DD *
    INCLUDE LOAD(TEMPNAME)
    ENTRY MAIN
//GO.FT10F001 DD UNIT=TAPE9,DISP=NEW,VOL=SER=410015,LABEL=(1,SL),
//     DCB=(DEN=3,RECFM=VBS,LRECL=11120,BLKSIZE=11124),
//  DSN=FILE1
//GO.FT10F002 DD UNIT=TAPE9,DISP=NEW,VOL=SER=410015,LABEL=(2,SL),
//     DCB=(DEN=3,RECFM=VBS,LRECL=11120,BLKSIZE=11124),
//  DSN=FILE2
//GO.FT10F003 DD UNIT=TAPE9,DISP=NEW,VOL=SER=410015,LABEL=(3,SL),
//     DCB=(DEN=3,RECFM=VBS,LRECL=11120,BLKSIZE=11124),
//  DSN=FILE3
//GO.FT10F004 DD UNIT=TAPE9,DISP=NEW,VOL=SER=410015,LABEL=(4,SL),
//     DCB=(DEN=3,RECFM=VBS,LRECL=11120,BLKSIZE=11124),
//  DSN=FILE4
//GO.FT10F005 DD UNIT=TAPE9,DISP=NEW,VOL=SER=410015,LABEL=(5,SL),
//     DCB=(DEN=3,RECFM=VBS,LRECL=11120,BLKSIZE=11124),
//  DSN=FILE5
//GO.FT10F006 DD UNIT=TAPE9,DISP=NEW,VOL=SER=410015,LABEL=(6,SL),
//     DCB=(DEN=3,RECFM=VBS,LRECL=11120,BLKSIZE=11124),
//  DSN=FILE6
//GO.FT10F007 DD UNIT=TAPE9,DISP=NEW,VOL=SER=410015,LABEL=(7,SL),
//     DCB=(DEN=3,RECFM=VBS,LRECL=11120,BLKSIZE=11124),
//  DSN=FILE7
//GO.FT10F008 DD UNIT=TAPE9,DISP=NEW,VOL=SER=410015,LABEL=(8,SL),
//     DCB=(DEN=3,RECFM=VBS,LRECL=11120,BLKSIZE=11124),
//  DSN=FILE8
//GO.FT11F001 DD DSN=ECMWF.CTLFIL,
// UNIT=3330,VOL=SER=MET043,DISP=OLD
//GO.FT12F001 DD DUMMY
//GO.FT13F001 DD UNIT=SYSDA,SPACE=(CYL,(31,1),RLSE),
//    DCB=(RECFM=VSB,BLKSIZE=13030)
//GO.FT14F001 DD DSN=ECMWF.TAPE14,
// UNIT=3330,VOL=SER=MET043,DISP=OLD
//GO.FT15F001 DD DSN=ECMWF.TAPE15,
// UNIT=3330,VOL=SER=MET043,DISP=OLD
//GO.FT16F001 DD DSN=ECMWF.IDISK,
// UNIT=3330,VOL=SER=MET043,DISP=OLD
//GO.FT19F001 DD DSN=ECMWF.ITAPE,
// UNIT=3330,VOL=SER=MET043,DISP=OLD
//GO.FT29F001 DD SYSOUT=A
//GO.SYSIN DD *
 $INPUT
  NRSTR=60,NTAUST=480,
  VREG=.FALSE.,NEWINT=.TRUE.,TYPE="DISK",
  IHIS=2100,ISMTH2=9999,NSMTH2=120,IDISP2=1,TYPESM="WAVG"
 $END
//
```

Table 4: JCL for a N24 run on the IBM 360/195

member	subroutines	member	subroutines	member	subroutines	member	subroutines
ANTL	ANTLG			LW1A	LW1	TR1A	TR1
ASMR	ASMRV	EXMA	EXMAIN	LW2A	LW2	TR2A	TR2
CALP	CALPHI	FACT	FACTR	GFDL	MAIN	TSCA	TSC
CDAT	CDATE	FIXR	FIXRL	MARC	MARK	TSCB	TSC1
CHKS	CHKSUM	FOUR	FOURTL	NEXD	NEXT	TSTA	TSTT
CLAS	BLOCK DATA	GOER	GOERT	NEPT	NEXTPT	TSTB	TST2
CLOU	CLOUD	HISW	HISWRT	NGMX	NGMXAD	UNMR	UNMRK
CLSF	CLSFY	HORZ	HORZDF	PRIN	PRINTR	VERT	VERTDF
COMP	COMPJD	HSRE	HSREAD	RADF	RADFLC	XINT	XINTGL
COND	CONDAD	IFMR	IFMRK	SMEQ	SMEQAD	ZENI	ZENITH
CONS	CONSOL	IMPD	IMPT	SMFA	SMFAC	CHANGE	JUMP, CORXFR,
CONT	CONST	INIT	INITLZ	SMSA	SMSC		TOD, TODALF,
CONV	CONVAD	INTE	INTERP	SOLA	SOLAR		DATE, GETIME,
COEL	COOL	INTR	INTRP	SRCA	SRC		JOBNAM
CPUM	CPUTIM	IPMA	IPM	STDP	STDPHI	ESATUR	ESATUR, ESDIFF,
DADA	DADADJ	LNGS	LNGSMF	SYMR	SYMRV		ESTABL.
ENDM	ENDTIM	LWFA	LWFD	TKOR	TCOR	BLOCKL	BLOCKL
ENDT	ENDTS	LWFB	LWFU	TRGF	TRGFC		

TABLE 2 Member names and corresponding subroutines of N24 version on the IBM 360/195.

FORTRAN FILE NO.	DSN	SPACE	DCB		
			RECFM	LRECL	BLKSIZE
10	ECMWF.HIST	(CYL,(40,1),, CONTIG)	VBS	11820	11824
11	ECMWF.CTLBLK	(TRK,(1))	VBS	404	408
14	ECMWF.TAPE14	(CYL,(31,1),, CONTIG)	FT	-	19520
15	ECMWF.TAPE15	(CYL,(31,1),, CONTIG)	FT	-	19520
18	ECMWF.IDISK	(CYL,(1,1))	VBS	328	332
19	ECMWF.ITAPE	(CYL,(1,1))	VBS	328	332

Table 3 Work disk files on disk METO43 for the N24 version

2. The N48 version on the MET. OFFICE IBM 360/195

2.1 Source code Modification from N24 version:

2.1.1 Fortran source modifications required to change the resolution

2.1.2 Additional routines to provide operator STOP facility and messages at history write time

2.1.3 Changes to the FFT package to prevent unnecessary repetition of calculations.

2.1.4 Replacement of existing Fortran routines by Assembler coded routines

2.1.2, 2.1.3 and 2.1.4 could readily be incorporated into the N24 version. All modifications except the assembler source for 2.1.2 and 2.1.4 are contained in ECMWF.SRCE.METO43 (GFDLN48M) as a set of IEBUPDTE control cards.

Details of changes:

ad 2.1.1 The following routines have changes:

BLOCK DATA	CLSFY	CONSOL	CONST
EXMAIN	HISWRT	HORZDF	HSREAD
IMPT	INITLZ	LNGSMF	MAIN
NEXT	NEXTPT	SRC	STDPHI
TRGFC	VERTDF	XINTGL	

ad 2.1.2 There are changes to MAIN, HISWRT, CONSOL and JUMP.

The STOP facility is implemented by inserting'CALL OPSTOP' in MAIN together with an additional common block / STOP / which contains the job name, stop indicator, plus the day and hour of the forecast (supplied by HISWRT) OPSTOP sends a message to the operator:
'JOBNAME REPLY STOP WHEN YOU WANT TO END THE JOB'.
An operator reply of 'STOP' sets the stop indicator in the common block which is interrogated by JUMP at the end of every time step.

.../41

ad 2.1.2 (continued)

The restart mechanism described in 1.7 is included in the CONSOL source modifications.
The routine TELLOP is called by HISWRT after completion of a history write sequence and it sends an operator message.
' JOBNAME DUMP AT Ddd Hhh'
where dd and hh are calculated from the number of time steps which have been executed, rounded to the nearest hour. The jobname is supplied to the common block STOP by the routine OPSTOP.
OPSTOP and TELLOP are coded in assembler and the source is stored in ECMWF.SRCE.MET043 (OPSTOP)
and " " " " (TELLOP)

ad 2.1.3

There are changes to FOURTL, COOL and FIXRL
In FOURTL, calls to SMFAC, SYMRV and ASMRV are avoided except for the first call to FOURTL.
In COOL, the calculations using DSIN are performed only once for each combination of factor and sign . In FIXRL, the calculations using DSIN are performed only once for each sign (ISIGN=$\pm$ 1)

ad 2.1.4

JOBNAM and CORXFR have been replaced by assembler coded routines.
JOBNAM retrieves the jobname as specified on the JOB card
CORXFR performs memory to memory transfers in the fastest possible way. The source is located in
ECMWF.SRCE.MET043 (JOBNAM)
and " " " (CORXFR)

.../42

2.2 Load modules

2.2.1 Fortran H compilation.

All modified routines were compiled under Fortran H (OPT=2) and included with ECMWF.LOAD.MET043 (TEMPNAME) to form with the assembler routines a new executable module called GFDLN48, located in the library ECMWF.LOAD on SYS002.

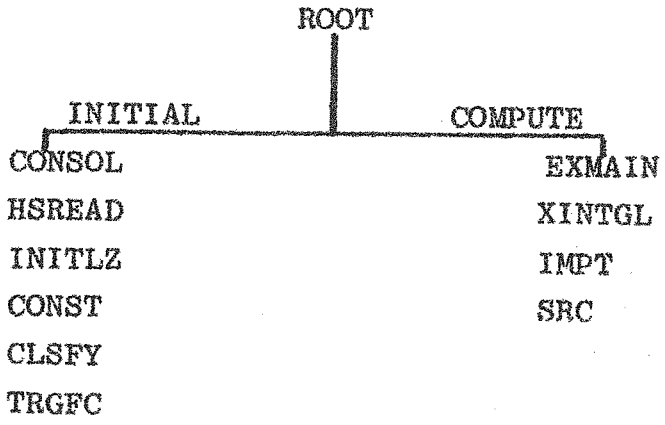
2.2.2 Fortran Extended H+ compilations.

The following routines have been recompiled using the Extended H+ compiler with OPT=3:-

*EXMAIN	*HORZDF	CALPHI	CONDAD	CONVAD	DADADJ
NGMXAD	SMEQAD	STDPHI	TRI	TR2	TSTT
TST2	ASMRV	FACTR	FIXRL	FOURTL	GOERT
SYMRV	CLSFY	COOL	IMPT	LNGSMF	MAIN
NEXT	NEXTPT	SRC	VERTDF	XINTGL	CHKSUM
CLOUD	LWFD	LWFU	LWI	LW2	RADFLC
TSC	TSC1	CONSOL	CONST		

* These were compiled with the SQRT function as inline code.

The compilations were combined with the load module GFDLN48 to produce a new module GFDLN48X on the same library.
All library functions were replaced by their extended H+ equivalent (IHC...) . There is a simple overlay structure to counteract memory overheads incurred by using the extended H+ compiler.



The root contains all routines not in the two overlay branches.

.../43

2.3 Memory and CPU requirements

More memory is needed when an integration is first started from initial conditions because of buffer requirements, etc. For subsequent restarts, a smaller region may be specified.

	H	EXT H+
START	558k	
RESTART	508k	
NORMAL TIME STEP	47sec	

Table 5 : Core requirements and normal time step length of N48 version on the IBM 360/195 (using two different compilers)

.../44

3. The N24 and N48 versions on the CDC 6600

3.1 Source Code

The original code that came from GFDL is on a permanent file SMAG,C4=9,FD=EWAH3. In its original form the code will not fit on the machine here and so it was modified. All the correction sets necessary to run it with resolutions of N24,N48, with/without overlay with/without double precision are on a PF FRED 2448, ID=EWAH3. For a particular run the user makes the appropriate modifications to the source, compiles it and catalogues the object file for use in all but the very first run.

3.2 Work Files

The model requires four work-files to be available at the beginning of a run. These are the (t-1,t) and (t,t+1) files on TAPE 14, TAPE 15, the control block on TAPE 11 and the accumulated integrals on TAPE 18. These are picked up at the beginning of the run and new cycles catalogued at the end. In the interim, if the user wishes to save the start up files they should be dumped to tape and purged from the disc as they are large.

3.3 History Files

N. Storer has developed a system to facilitate the tape management for the history files. A table is prepared which tells which tapes may be written to and how many have been filled already. When the model writes a history file it also initiates a new job. This job looks up the VSN TABLE to find which is the next tape to be written to and calls on the operator to mount the tape. When this is done a second job is automatically started which writes the history file to tape, updates the VSN TABLES and purges the history from disc This whole system can be disabled and the history accumulated as permanent files on disc if required. This is done for runs when there is no operator but lots of disc space is available e.g. overnight or over a week-end.

3.4 Overlays

There is a feature in the code which reduces the core requirement to the minimum required to run when the start-up, radiation, or history writing overlays are not in use.

3.5 Fortran Trap

From time to time the model can fail if another user demands a lot of disc space. This would not disturb the running of the model if it used random access to the disc. In such a case if the files are closed properly they can be used and no CP time is lost. There is a feature in the code to trap the Fortran error, allowing the files to be properly closed before job termination.

3.6 Stopping a run

A run is stopped by setting SWITCH 1 on the Console.

3.7 Examples

Table 5 shows the day file of an initial run with cataloguing of the OBJECT module.

Table 6 shows the day file of a normal run, using the object module created in the previous run.

```
  MFA    ECMWF SN18 6613 CMR 1 L 410    19/07/76
09.44.19.EWAH307  FROM
09.44.19.IP  00001600 WORDS - FILE INPUT  , DC 00
09.44.19.EWAH3,T0,NT1.          CREATE N48INTRAD BI
09.44.19.NARY DECK
09.44.20.MAP,PART.
09.44.20.ATTACH,TAPE10,HISTORYN48,ID=EWAH3,RW=1,M
09.44.20.R=1,PW=*---*,*---*.
09.44.20.PF CYCLE NO. = 001
09.44.20.REQUEST,TAPE14,*SN=SYSSET,VSN=ECMWF5.
09.44.20.REQUEST,TAPE15,*SN=SYSSET,VSN=ECMWF6.
09.44.20.REQUEST,TAPE11,*PF.
09.44.20.REQUEST,TAPE18,*PF.
09.44.21.ATTACH,FRED,FRED2448,ID=EWAH3,MR=1.
09.44.21.PF CYCLE NO. = 001
09.44.21.ATTACH,OLDPL,SMAG,ID=EWAH3,CY=9,MR=1.
09.44.21.UPDATE(F,*==,D,8,C=IN,P=FRED,L=F.
09.44.24.          1 OVERLAPPING CORRECTIONS
09.44.26. UPDATE COMPLETE.
09.44.26.UPDATE,F,P,I=IN,N=ANTO,L=A124.
09.44.33.DECK STRUCTURE CHANGED
09.45.03.          6 OVERLAPPING CORRECTIONS
09.45.04. UPDATE COMPLETE.
09.45.04.REQUEST,LGO,*PF.
09.45.04.FTN,A,L=0,I=COMPILE,OPT=2.
09.45.07.      DEAD CODE IN - MAIN
09.47.19.      DEAD CODE IN - HSREAD
09.48.11.     62.695 CP SECONDS COMPILATION TIME
09.48.11.RETURN(OLDPL,FRED,IN,COMPILE)
09.48.12.ATTACH(OLDPL,ROUTERUPS,ID=EWNS1)
09.48.12.PF CYCLE NO. = 001
09.48.12.UPDATE(Q)
09.48.13. UPDATE COMPLETE.
09.48.13.RETURN(OLDPL)
09.48.13.RFL(77000)
09.48.13.FTN,A,L,I=COMPILE,OPT=2,S=PFMTEXT,S=SCPT
09.48.13.EXT)
09.48.20.      2.554 CP SECONDS COMPILATION TIME
09.48.20.REDUCE.
09.48.20.RETURN(COMPILE)
09.48.20.LABEL,TAPE22,R,D=PE,NORING,VSN=EW0124.
09.49.46.( NT20 ASSIGNED)
09.49.46.NT20 VOLUME SERIAL NUMBER IS  EW0124
09.49.48. LABEL READ WAS      N48INTRADVOL02
09.49.48.    EDITION NUMBER    01
09.49.48.    RETENTION CYCLE   000
09.49.48.    CREATION DATE     76167
09.49.48.    REEL NUMBER       0001
09.49.48.COPYBR,TAPE22,TAPE12.
09.51.08.REWIND,TAPE12.
09.51.08.UNLOAD,TAPE22.
09.51.08.CATALOG,LGO,N48OBJINTRAD,ID=EWAH3.
09.51.09.NEWCYCLE CATALOG
09.51.09.PF CYCLE NO. = 005
09.51.09.RP = 005 DAYS
09.51.09.LOAD,LGO.
09.51.10.NOGO.
09.51.18.      NON-FATAL LOADER ERRORS - SEE MAP
09.51.30.      NON-FATAL LOADER ERRORS - SEE MAP
09.51.30.OVER.
09.51.39.
10.03.41.*=*=*=*=* CM SIZE NOW *=*=*=*=*
10.03.41.          00002711008
```

table 5. Day file of an initial N48 run on the CDC 6600 with CATALOG of the object module

Table 5: Day file of an initial N48 run on the CDC 6600 with CATALOG of the object module

```
 MFA   ECMWF SN18 6613 CMR 6 L 410   26/08/76
08.36.59.EWAH365  FROM
08.36.59.IP  JOJ00384 WORDS - FILE INPUT  , DC 60
08.36.59.EWAH3,T0,P3.  START FROM FILES ON DISC.
08.36.59.
08.36.59.MAP(PART)
08.37.00.ATTACH,TAPE10,HISTORYN48,ID=EWAH3,RW=1,M
08.37.00.R=1,PW=*---*,*---*.
08.37.00.PF CYCLE NO. = 001
08.37.00.REQUEST,TAPE14,*SN=SYSSET,VSN=ECMWF7.
08.37.00.REQUEST,TAPE15,*SN=SYSSET,VSN=ECMWF8.
08.37.00.REQUEST,TAPE11,*PF.
08.37.00.REQUEST,TAPE18,*PF.
08.37.00.ATTACH,TAPE21,CTLBLK06JUN76,ID=EWAH3.
08.37.01.PF CYCLE NO. = 002
08.37.01.COPY,TAPE21,TAPE11.
08.37.01.ATTACH,TAPE24,TAPE1406JUN76,ID=EWAH3.
08.37.01.PF CYCLE NO. = 002
08.37.01.COPY,TAPE24,TAPE14.
08.38.30.ATTACH,TAPE25,TAPE1506JUN76,ID=EWAH3.
08.38.31.PF CYCLE NO. = 002
08.38.31.COPY,TAPE25,TAPE15.
08.39.53.ATTACH,TAPE28,INTGRL06JUN76,ID=EWAH3.
08.39.54.PF CYCLE NO. = 006
08.39.54.COPY,TAPE28,TAPE18.
08.40.08.REWIND,TAPE11,TAPE14,TAPE15.
08.40.08.RETURN,TAPE21,TAPE24,TAPE25,TAPE28.
08.40.09.ATTACH,LGO,N48OBJINTRAD,ID=EWAH3,MR=1.
08.40.09.PF CYCLE NO. = 006
08.40.09.LOAD,LGO.
08.40.09.NOGO.
08.40.17.      NON-FATAL LOADER ERRORS - SEE MAP
08.40.27.      NON-FATAL LOADER ERRORS - SEE MAP
08.40.27.OVER(PL=99999)
08.49.15.*=*=*=*=* CM SIZE NOW *=*=*=*=*
08.49.15.          0000271100B
10.54.41.*=*=*=*=* CM SIZE NOW *=*=*=*=*
10.54.41.          0000323300B
11.07.06.*=*=*=*=* CM SIZE NOW *=*=*=*=*
11.07.06.          0000271100B
11.07.06.*=*=*=*=* CM SIZE NOW *=*=*=*=*
11.07.06.          0000323300B
11.07.43.
11.21.52.*=*=*=*=* CM SIZE NOW *=*=*=*=*
11.21.52.          0000271100B
11.38.09.LOCKOUT.
11.41.49.UNLOCK.
11.54.27.LOCKOUT.
11.54.50.UNLOCK.
12.51.53.LOCKOUT.
12.51.53.UNLOCK.
13.10.05.LOCKIN.
13.10.23.UNLOCK.
16.18.47.LOCKOUT.
17.13.13.UNLOCK.
17.13.16.LOCKOUT.
17.32.11.UNLOCK.
17.32.17.ONSW1.
17.39.37.LOCKIN.
17.41.03.    STOP  77777
17.41.03.  9092.052 CP SECONDS EXECUTION TIME
17.41.03.EXIT,U.
17.41.03.DISPOSE,TAPE29,PR.
17.41.03.OP  J0000256 WORDS - FILE TAPE29 , DC 40
17.41.03.CATALOG,TAPE11,CTLBLK06JUN76,ID=EWAH3.
```

Table 6. Day file of a normal Nu8 run, using the object module created in the previous run (is continued on next page)

/continued .....

Table 6:

Continuation of Table 6:

```
17.41.03.INITIAL CATALOG
17.41.03.PF CYCLE NO. = 001
17.41.04.RP = 005 DAYS
17.41.04.CATALOG,TAPE13,INTGRL06JUN76,ID=EWAH3.
17.41.04.INITIAL CATALOG
17.41.04.PF CYCLE NO. = 001
17.41.04.RP = 005 DAYS
17.41.04.CATALOG,TAPE14,TAPE1406JUN76,ID=EWAH3.
17.41.04.INITIAL CATALOG
17.41.04.PF CYCLE NO. = 001
17.41.05.RP = 005 DAYS
17.41.06.CATALOG,TAPE15,TAPE1506JUN76,ID=EWAH3.
17.41.06.INITIAL CATALOG
17.41.06.PF CYCLE NO. = 001
17.41.07.RP = 005 DAYS
17.41.08.AUDIT,ID=EWAH3.
17.41.18.  EXIT
17.41.18.OP  30034816 WORDS - FILE OUTPUT , DC 40
17.41.18.MS 4182528  WORDS  ( 4182528 MAX USED)
17.41.18.CPA    9100.812 SEC.      9100.812 ADJ.
17.41.18.IO     2490.242 SEC.      2490.242 ADJ.
17.41.18.CM 1107456.381 KWS.     67593.773 ADJ.
17.41.18.SS                       79184.827
17.41.18.PP     2721.622 SEC.      DATE 09/09/76
17.41.18.EJ  END OF JOB, **

**********          EWAH365  //// END OF LIST ////
**********          EWAH365  //// END OF LIST ////
```

Table 6: Day file of a normal N48 run, using the object module created in the previous run.

APPENDIX

NOTES FOR THE OPERATORS HANDLING G F D L MODEL RUNS

1. Introduction

The GFDL model uses a number of permanent files for its integrations. The main Work Files are called tape 14 and tape 15. At any instant it is reading from one of these files, producing a new forecast and writing it to the other. At the end of a time step the read file becomes the write file and vice versa. These two files, together with tape 11 and tape 18 very short files, catalogued as CTLBLK (for control block) and INTGRL contain all the data necessary to run or restart the model.

When the model is running it can be stopped by doing ONSW1., i.e. on-switch - 1. This causes the model to stop within a few minutes, at the end of the current time step. It then catalogues its work files Tape 14, Tape 15 and Tape 11 and Tape 18, which are therefore available for restart.

From time to time we must make a record of the state arrived at by the model. This is referred to as writing a history tape. N. Storer has written a suite of programs which facilitate this procedure. When the history file is ready to be copied to tape, it is catalogued and two new jobs are initiated. One informs the operator of the tape required. The second job copies the file to tape, once the tape is mounted. If the copy is successful the job also purges the history file.

...2

NOTES FOR THE OPERATORS HANDLING GFDL MODEL RUNS

Thus if everything is running smoothly the system works as follows : At the beginning of a session there are on catalogued disc file two cycles each of Tape 14, Tape 15, CTLBLK, INTGRL. The low cycles were used to start the previous day's run and the high cycles were catalogued at the end of the previous day's run. The disc restart deck is read in, the model picks up the high cycles and resumes its run. When it is clear that the restart worked properly the low cycles may be purged. There is a 6-card deck provided for this purpose. From time to time during the session the operator will be asked to mount a tape so that the history can be saved. When the operator wishes to discontinue the run he sets switch 1, the job finishes within two or three minutes. It also catalogues its work files as new high cycles, ready for the next day's run.

Problems that arise.

The code is well tested so that genuine errors within the code are rare. The commonest problems arise through running out of disc space, hardware faults on the main frame, disc hardware problems, failure of the copy of the history to tape.

1. Running out of disc space

Depending precisely on how the job terminates it may or may not produce new high cycles of its work files and these may or may not be usable. If it fails through a Fortran error then it will always produce new high cycles. Running out of disc space causes a Fortran error. If this is identified as the cause of the problem then the high cycles may be used to restart the job in the normal way, once space is available on the disc.

2. Hardware faults

These are sometimes hard to identify on first encounter. For example the job has failed with a Fortran error message " SQRT with negative argument" when it was evaluating SQRT (x^2+y^2), x,y real. Fortran errors always result in new high cycles being catalogued. If a hardware error is suspected in such a case then one should purge the high cycles and restart from the low cycles.

Sometimes, however, in a case of hardware failure, no Fortran Error Message is issued and the run terminates with no new high cycles being catalogued. If it seems likely that the fault is in the hardware then the run may be restarted from the existing cycles.

3. Disc hardware problems

Sometimes it may happen that the files on the disc are unusable. When this sad situation is reached we must restart from the most recent history tape. We describe below the details of the writing of the history tapes. For the moment it suffices that it is easy to establish the tape numer, tape label, and file number on the tape of the most recent history file. This information must be given to the history restart deck in the form of a label card and a SKIPF card. In addition the time step number of the point from which we are restarting must be provided to the namelist input of the history restart deck in the form MRSTR=nnnn . This information is also readily available from the table of history tapes.

General

Suppose that a run fails in the course of a session, suppose further that the above notes indicate that the restart should be made from the disc files used to start the day's work and suppose finally that a history tape has been written since the morning start and that there is every reason to believe that the history is o.k. In these circumstances one should, for reasons of economy, restart from the history tape.

Problems with the history tapes

The system for writing the history tapes works as follows: There is a small permanent file called VSN TABLES which has a record of the tape numbers and labels of the tapes reserved for the history. It also tells how many files, if any, are written on each tape together with information about the date and time when each file was written together with the time step number of the file. When a history file is ready to be dumped to tape the model initiates two small jobs. The first job catalogues the history file and sets a flag on a permanent file HISTFLAG. If the flag is already set then two or more files must be copied. At any rate the job looks up the VSN TABLES and issues a request to the operator to mount the appropriate tape.

The second job begins when the tape is mounted. The tape is positioned at the end of information and the history copies from disc to tape. When the copy is successful the VSN TABLES are updated, the HISTFLAG is reset to null, the history on disc is purged and the tape unloaded.

Problems

If for any reason this process breaks down the following procedure should be followed :

Job 1 Attach this history file on disc, itemize it to find how many files are present.

Job 2 Look up the printed VSN TABLES to find the next tape to be written to and copy the disc to tape.

Job 3 From the printed output of the main job determine the details of the history files and update the VSN tables

Job 4 Reset the HISTFLAG

Job 5 Purge the history on disc by attaching the file, then doing REWIND followed by ALTER. This leaves a file of zero length on the disc.

Decks for each of these jobs will be provided but the operators must insert tape numbers and labels, number of files to be skipped in SKIPF instructions, and the data for the updating of the VSN TABLES.

3.6.76

AH/AILD