JAERI-M
6251

MODIFICATION OF THE MORSE CODE FOR MONTE CARLO EIGENVALUE PROBLEMS BY COARSE MESH REBALANCE ACCELERATION

September 1975

T.NISHIDA, K. HORIKAMI, T. SUZUKI,
Y. NAKAHARA, Y. TAJI, and T. ASAOKA

日本原子力研究所
Japan Atomic Energy Research Institute

この報告書は，日本原子力研究所がJAERI-Mレポートとして，不定期に刊行している研究報告書です．入手，複製などのお問合わせは，日本原子力研究所技術情報部（茨城県那珂郡東海村）あて，お申しこしください．

JAERI-M reports, issued irregularly, describe the results of research works carried out in JAERI. Inquiries about the availability of reports and their reproduction should be addressed to Division of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken, Japan.

JAERI-M 6251

Modification of the MORSE Code for Monte Carlo Eigenvalue Problems by Coarse-Mesh Rebalance Acceleration

Takahiko NISHIDA, Kunihiko HORIKAMI, Tadakazu SUZUKI,
Yasuaki NAKAHARA, Yukichi TAJI and Takumi ASAOKA

(Received September 5, 1975)

The coarse-mesh rebalancing technique is introduced into the general-purpose neutron and gamma-ray Monte Carlo transport code MORSE, to accelerate the convergence rate of the iteration process for eigenvalue calculation in a nuclear reactor system. Two subroutines are thus attached to the code. One is bookkeeping routine 'COARSE' for obtaining the quantities related with the neutron balance in each coarse mesh cell, such as the number of neutrons absorbed in the cell, from random walks of neutrons in a batch. The other is rebalance factor calculation routine 'REBAL' for obtaining the scaling factor whereby the neutron flux in the cell is multiplied to attain the neutron balance. The two subroutines and algorithm of the coarse mesh rebalancing acceleration in a Monte Carlo game are described.

JAERI－M 6251

粗メッシュ再鈎合い加速法によるモンテカルロ，コード MORSE の改良

日本原子力研究所東海研究所原子炉工学部
西田雄彦　掘上邦彦　鈴木忠和　中原康明
田次邑吉　朝岡卓見

（１９７５年９月５日受理）

原子炉系の固有値を計算する際，その繰返し計算過程の収束を加速するために，汎用モンテカルロ輸送コード“MORSE”に粗メッシュ再鈎合い法を適用した。このために新たに２つのサブルーチンが従来の“MORSE”に付け加えられた。その１つは，モンテカルロ・ゲームの１バッチ・ランに含まれるすべてのランダム・ウォークから，中性子の吸収数など，各粗メッシュ・セル中の中性子バランスに関連した諸量を集積処理するルーチン“COARSE”である。もう一方はこれらの量から，中性子バランス達成するために中性子束に乗せられるスケール・ファクターを計算するルーチンの“REBAL”である。ここでは，この２つのルーチンの詳細を述べると共に，モンテカルロ・ゲームにおいて粗メッシュ再鈎合い加速法のアルゴリズムについて報告する。

CONTENTS

≪ 目　　次 ≫

1. Introduction

The Monte Carlo method has recently taken increasingly an important post in a wide field of theoretical study in reactor physics as computers grow in their abilities. However, the problem exists still in the fact that it takes too much computer time for obtaining the result with a small statistical error, especially for solving eigenvalue problems. One of the urgent necessities is therefore to accelerate the convergence rate of iterations of Monte Carlo eigenvalue calculations.

We have shown in a previous report[1] that the coarse mesh rebalancing method can successfully be used for accelerating the convergence of one dimensional eigenvalue calculations. At every completion of the Monte Carlo game for one batch of neutron histories, the scaling factor for the neutron flux is calculated so as to achieve the neutron balance in each coarse mesh cell. The rebalancing factor is multiplied to the weight of each fission neutron in the coarse mesh cell for playing the next batch of sampling calculations.

This acceleration tachnique has recently been extended to two dimensional calculations[2]. The results have shown that the rebalancing method gives a new usable sampling technique for the estimation of the number of neutrons lost or produced in each coarse mesh cell.

In order to apply the coarse mesh rebalancing method to Monte Carlo calculations, we have programmed two new subroutines "COARSE" and "REBAL", and added those to "MORSE"[3]. On solving one, two or three dimensional transport equation for neutrons in a reactor, the rebalancing equation in a coarse mesh cell L may be written as

$$f_L\left[\sum_{L'}(CUR)_{L\to L'} + (AB)_L + (SD)_L\right] = (QQ)_L + \sum_K f_K\cdot(CUR)_{K\to L}, \quad \cdots\cdots [1\text{-}1]$$

where
- $(CUR)_{L\to L'}$: total neutron current crossing the interface from a cell L to a cell L',
- $(AB)_L$: the number of neutrons absorbed in a cell L,
- $(SD)_L$: total number of neutrons slowing down below the cut-off energy in a cell L,
- $(QQ)_L$: total number of source neutrons for a celll L, $= f_L\cdot\frac{(FS)_L}{k_{eff}}$,
- $(FS)_L$: total number of fission neutrons produced in a cell,
- f_L : rebalance factor for a cell L,

k_{eff} : effective multiplication factor.

..... [1-2]

One of the new subroutines, COARSE gets informations of neutron absorption, boundary crossing, slowing down and fission from random walks in "MORSE". Another subroutine "REBAL" calculates the rebalance factor f_L using the output data from the "COARSE".

The present report describes the details of these two subroutines as well as the algorithms for solving equation [1-1].

2. Bookkeeping subroutine "COARSE"

Let f_L be the rebalance factor for a cell L and W_{hj} neutron's weight at the j-th collision on the h-th history. Then, each term in equation [1-1] is given by the following "weight summation" in a Monte Carlo game:

$$(CUR)_{L\to L'} = \sum_h \sum_j W_{h,j-1}^{L\to L'} \quad \text{.....} \quad [2\text{-}1]$$

$$(AB)_L = \sum_h \sum_j \left(\frac{\Sigma_a}{\Sigma_t}\right)_L^g \cdot W_{h,j-1} \quad \text{.....} \quad [2\text{-}2]$$

$$(SD)_L = \sum_h W_{hJ} \quad \text{.....} \quad [2\text{-}3]$$

$$(FS)_L = \sum_h \sum_j \left(\frac{\nu\Sigma_f}{\Sigma_t}\right)_L^g \cdot W_{h,j} \quad \text{.....} \quad [2\text{-}4]$$

where Σ_a, Σ_f, Σ_t and ν designate, respectively, the absorption cross section, fission cross section, total cross section and average number of neutrons emitted per fission. The g is the energy group index and J the number of collisions at which a neutron slows down below the cut-off energy.

The present routine is the pre-processor routine which accepts the random walk information and calculates the quantities [2-1] ∿ [2-4] for the rebalance factor calculation. It is known that the rebalancing method with coarse mesh cells effectively accelerates the convergence of iterative processes of the current S_N computer codes. However, when the method is especially applied to a Monte Carlo calculation, it is necessary to accumulate a great deal of informations of neutron histories for obtaining accurate values of f_L's. In our program a coarse mesh cell corresponds to one medium region, whose material composition is uniform in "MORSE", and f_L's are computed at the end of one batch run.

Therefore, $(CUR)_{L\to L'}$, $(AB)_L$, $(SD)_L$ and $(FS)_L$ have to be obtained from the summation over histories of one batch on a medium cell L. As seen in

k_{eff} : effective multiplication factor.

..... [1-2]

One of the new subroutines, COARSE gets informations of neutron absorption, boundary crossing, slowing down and fission from random walks in "MORSE". Another subroutine "REBAL" calculates the rebalance factor f_L using the output data from the "COARSE".

The present report describes the details of these two subroutines as well as the algorithms for solving equation [1-1].

2. Bookkeeping subroutine "COARSE"

Let f_L be the rebalance factor for a cell L and W_{hj} neutron's weight at the j-th collision on the h-th history. Then, each term in equation [1-1] is given by the following "weight summation" in a Monte Carlo game:

$$(CUR)_{L\to L'} = \sum_h \sum_j W_{h,j-1}^{L\to L'} \quad \text{.....} \quad [2\text{-}1]$$

$$(AB)_L = \sum_h \sum_j \left(\frac{\Sigma_a}{\Sigma_t}\right)_L^g \cdot W_{h,j-1} \quad \text{.....} \quad [2\text{-}2]$$

$$(SD)_L = \sum_h W_{hJ} \quad \text{.....} \quad [2\text{-}3]$$

$$(FS)_L = \sum_h \sum_j \left(\frac{\nu\Sigma_f}{\Sigma_t}\right)_L^g \cdot W_{h,j} \quad \text{.....} \quad [2\text{-}4]$$

where Σ_a, Σ_f, Σ_t and ν designate, respectively, the absorption cross section, fission cross section, total cross section and average number of neutrons emitted per fission. The g is the energy group index and J the number of collisions at which a neutron slows down below the cut-off energy.

The present routine is the pre-processor routine which accepts the random walk information and calculates the quantities [2-1] ∿ [2-4] for the rebalance factor calculation. It is known that the rebalancing method with coarse mesh cells effectively accelerates the convergence of iterative processes of the current S_N computer codes. However, when the method is especially applied to a Monte Carlo calculation, it is necessary to accumulate a great deal of informations of neutron histories for obtaining accurate values of f_L's. In our program a coarse mesh cell corresponds to one medium region, whose material composition is uniform in "MORSE", and f_L's are computed at the end of one batch run.

Therefore, $(CUR)_{L\to L'}$, $(AB)_L$, $(SD)_L$ and $(FS)_L$ have to be obtained from the summation over histories of one batch on a medium cell L. As seen in

expressions [2-1] ∿ [2-4], different energy-groups are involved in each event of random walk but the contributions have already been summed up in a Monte Carlo game. Generally speaking, the coarse mesh rebalancing method accelerates effectively the convergence only at the beginning of iterative processes but its effect slows down after a few times application. Therefore, the new modified "MORSE" need also have the normal Monte Carlo option without the coarse mesh rebalancing not to waste computing time.

The neutron current into a cell $L^{(*)}$ or from a cell L in the rebalancing method is given by summing up the weight of neutrons passing the cell boundary over all histories in one batch. Hence, for simplicity of matrix calculation to obtain a rebalance factor f_L, the cell structure is assumed only of rectangular type. Especially in two dimensional geometry, the current is decomposed to the component on each side of a lattice-like cell.

2.1 COARSE for one dimensional geometry

As shown in Fig. 1-(a), we define FRX_ℓ the current from a cell $(\ell-1)$ to ℓ and inversely FLX_ℓ the current from ℓ to $(\ell-1)$. The $FRX_{\ell+1}$ and $FLX_{\ell+1}$ are defined similarly on the boundary between cells ℓ and $\ell+1$. Therefore, two terms including the current in the balance equation [1-1] are rewritten as

$$\left.\begin{aligned} \sum_{\ell'} (CUR)_{\ell\to\ell'} &= FRX_{\ell+1} + FLX_\ell, \\ \sum_k f_k (CUR)_{k\to\ell} &= f_{\ell-1}\cdot FRX_\ell + f_{\ell+1}\, FLX_{\ell+1} \end{aligned}\right\}, \quad \cdots\cdots \quad [2\text{-}5]$$

where FRX_1, FLX_1 and $FRX_{(\ell+1)max}$ and $FLX_{(\ell+1)max}$ depend on the boundary conditions. On the vacuum boundary, for example, FLX_1 or $FRX_{(\ell+1)max}$ gives the net loss of neutrons from the whole system, and FRX_1 or $FLX_{(\ell+1)max}$ becomes zero.

In addition as shown in Fig. 1-(b) it is possible to divide the uniform medium region (A) into cells (1)∿(3) to obtain more rapid acceleration of the convergence by this method in the modified MORSE. In this case, each cell in the same medium has to be given the same total cross sections for the actual calculation of collision. The subroutine for one dimensional rebalancing method is named as "COARSE", the details of which are given in APPENDICES I and IV.

(*) L represents general index of a coarse mesh cell. So in one dimension L denotes ℓ and in two dimension (ℓ, ℓ').

2.2 COARS2 and COARS3 for two dimensional geometry (cylinder)

One can extend naturally the procedure mentioned above for one dimensional case to two dimensional case. For calculating the neutron current, the weight of a neutron passing on each boundary of a coarse mesh cell, which is assigned by two indices [ℓ, ℓ'] in this case, is summed up to obtain four quantities FRX, FLX, FRY and FLY, as indicated in Fig.2-(a). For example, on the right boundary of Fig. 2-(a), the current from [ℓ+1,ℓ'] to [ℓ,ℓ'] is stored in FLX(ℓ+1,ℓ') and the opposite current in FRX(ℓ+1,ℓ'). As seen in Fig. 2-(a), the index for the absorption (AB), fission (FS) or slowing down (SD) is in agreement to the cell index or the index of the current on the left or lower boundary of the cell [see also Fig. 2-(b)].

Therefore, the expressions in [2-5] are, in two dimensional geometry, written in the forms:

$$\sum_{L'} (CUR)_{L\to L'} = FRX(\ell+1,\ell')+FLX(\ell,\ell')+FRY(\ell,\ell'+1)+FLY(\ell,\ell'),$$

$$\sum_{K} f_K (CUR)_{K\to L} = f_{\ell-1,\ell'}\cdot FRX(\ell,\ell')+f_{\ell,\ell'-1}\cdot FLY(\ell,\ell') + f_{\ell+1,\ell'}\cdot FLX(\ell+1,\ell')+f_{\ell,\ell'+1}\cdot FLY(\ell,\ell'+1).$$

..... [2-6]

When one desires to calculate the total leakage TCR from the outer boundaries (the rightest and highest boundaries), the following formula is used:

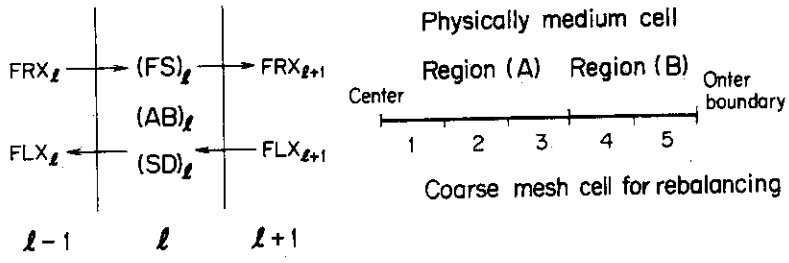
$$TCR = \sum_{MY=1}^{IRY} FRX(IRXP,MY) + \sum_{MX=1}^{IRX} FRY(MX,IRYP) ,$$

where IRX and IRY are the total mesh numbers in r- and Z-directions respectively (IRXP=IRX+1, IRYP=IRY+1).

The subroutine for two dimensional rebalancing method is named as "COARS2". Furthermore, for a parctical use, the "COARS2" is extended to "COARS3", in which meshes in the z-direction have to form perfectly cross sectional boundaries, while those in the r-direction can be taken arbitrarily for each z-mesh (see Fig. 3). The flow diagram of "COARS2" is shown in APPENDIX II and the program lists of "COARS2" and "COARS3" are given in APPENDIX IV.

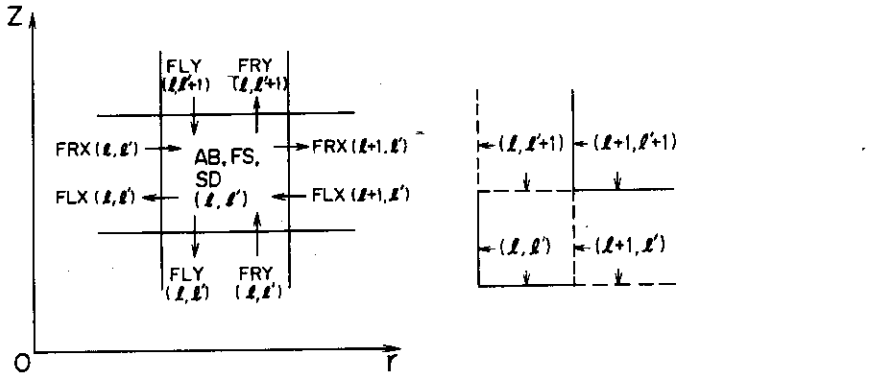
2.3 Interface with "MORSE"

The "MOSE" has many options of storing various informations obtained



1 — (a) 1 — (b)

Fig. 1 One dimensional geometry and coarse mesh cell



2 — (a) 2 — (b)

Fig. 2 Neutron current for two dimensional cell

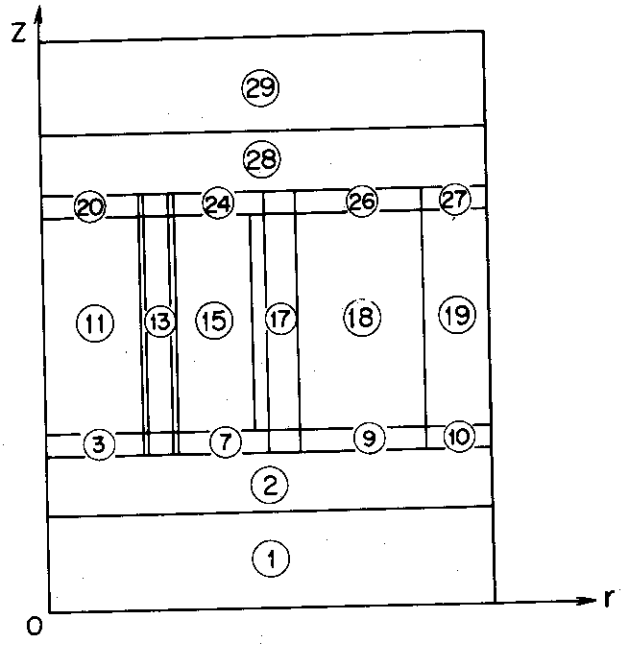


Fig. 3 Twentynine-zone cylindrical fast reactor model

from the Monte Carlo game in the form of the so-called collision tape[4]. Those informations, for example, the weight of neutrons in an event on each random walk, are controlled only by the subroutine "BANKR". After the occurrence of any event on random walk, all the necessary informations generated on the event are stored by "BANKR" through the labeled COMMONs /NUTRON/ and /FISBNK/.

In the present rebalance options, "COARSE" in modified "MORSE" is immediately called after the entry to "BANKR" as shown in Fig. 6 and it obtains the data necessary for the coarse mesh rebalancing over all histories in one batch. These data are as follows:

MEDOLD : index of a medium (=one coarse mesh cell) where there exists the previous collision point,

NMED : index of a medium where there exists a new collision point,

WATE : weight of a particle (=a neutron) at a new collision point,

OLDWT : weight of a particle at the previous collision point,

WATEF : fission weight at a new collision point, defined by WATE*PNUF, where PNUF = $(\frac{\nu\Sigma f}{\Sigma t})_L^g$,

IG : index of energy group of a particle at a new collision point, which is required for the COARSE to obtain the next parameter PNAB,

PNAB : non-absorption probability, which is a function of medium number and group number at a collision point. Since it is exceptionally not included in COMMON bank, we have to calculate the value using the subroutine NSIGTA.

As three dimensional co-ordinates of a collision point are included in COMMON bank, we can know the relation between a cell boundary plane and the direction of the particle movement. In our method described above, however, it is not necessary to obtain the angle with which a particle crosses on the boundary plane but only the total weight of particles passing the whole boundary plane. Even if a particle passes through a cell without collision, the cell number is replaced by the next cell number by taking an imaginary collision point on the cell boundary. So in the process of summation, neutrons coming only from the neighboring cells

may be taken account for obtaining the current. Events required for the present method out of many events of "MORSE" and operation in "COARSE" for the events are summarized in Table 1. For the event 8, "MORSE" gives a signal NMED=0. At the corner cell ABCD in two dimensional geometry shown in Fig. 4, we cannot determine from NMED=0 whether a particle escapes through the boundary AD or CD. However, as these components of leakage are added together to the first term in parentheses on the left hand side of the balance equation [1-1], it is not necessary to distinguish the leakage through AD from that through CD. We therefore set in the "COARS2" that the leakage through AD is equal to that through CD, and in the "COARS3" that these values are proportional to the surface areas.

The modification performed on the "MORSE" for introducing the coarse mesh rebalance option is summarized as follows:

(1) A new input IREBAL is added as the fourteenth input of the second card B read by the subroutine "INPUT" with FORMAT(14I5);
IREBAL = 0; don't carry out the coarse mesh rebalancing,
= n; apply the rebalancing to the first n batches.

(2) Two new labeled COMMON's are added;
/COARS/NBATCH, IREBAL, SWATE,
where NBATCH is the batch number and SWATE is the sum of source particle weights.
/FACTOR/RFACT(L),
where

$$\mathrm{RFACT}(L) = \prod_{n} f_L^{(n)} \cdot \frac{k_{eff}^{(0)}}{k_{eff}^{(\mathrm{newest})}} \quad ,$$

which is multiplied to the weight of each fission neutron produced in the cell L, stored in the fission bank of the blank COMMON (see the next section).

(3) Slight alteration of the subroutine "BANKR" as shown in Fig. 6, and the subroutine "SOURCE" to multiply RFACT(L) to the weight of each fission neutron source.

(4) Restriction;
Cell numbers (MED) in two dimensional case have to be sequentially named radially from the center, with the outer loop from the bottom in the z-direction (see Fig. 4).

Table 1 Sampled events for the coarse mesh rebalance calculation

Event index	Event	Operation in COARSE
-2	batch start	·Initialize FRX^L and FLX^L
-3	batch end	·Output FRX^L and FLX^L CALL REBAL
3	fission	·Add WATEF(h,j) to $(FS)_L$
5	real collision	·CALL NSIGTA to obtain PNAB ·Add OLDWT(h,j)*(1-PNAB) to $(AB)_L$
7	boundary crossing	·Add OLDWT(h,j) of a particle crossing the boundary of a medium cell to the corresponding counter for the current with taking the direction into consideration
8	escape (NMED=0)	·Add OLDWT(h,j) to the counter of the outward current on the surface
9	energy cut-off	·Add WATE(h,J) to $(SD)_\ell$

(*) Index (h,j) indicates the j-th collision in the h-th history.

3. Rebalance factor calculation subroutine "REBAL"

Whenever all calculations on histories of particles included in one batch have completed, the "REBAL" can obtain the information necessary to calculate the rebalancing factor f_L from the "COARSE". This section describes the method to solve the balance equation [1-1] for obtaining f_L's with the use of the quantities obtained in Monte Carlo game, the boundary condition and convergence criterion for one, two and three dimensional cases. However, in APPENDIX, the program flow chart is given only for one dimensional subroutine "REBAL", because it is essentially the same for the other cases.

The balance equation [1-1] on a lattice like cell is represented for each case as follows (see Fig. 1, Fig. 2 and Fig. 5);
for one dimensional case,

$$f_i[FLX_i+FRX_{i+1}+AB_i+SD_i] = QQ_i+f_{i-1}FRX_i+f_{i+1}FLX_{i+1},$$

$$(i=1 \sim IX) \quad \ldots\ldots \quad [3\text{-}1]$$

for two dimensional case,

$$f_{i,j}[FLX_{i,j}+FRX_{i+1,j}+FLY_{i,j}+FRY_{i,j+1}+AB_{i,j}+SD_{ij}]$$

$$= QQ_{i,j}+f_{i-1,j}\cdot FRX_{i,j}+f_{i+1,j}\cdot FLX_{i+1,j}+f_{i,j-1}\cdot FRY_{i,j}+f_{i,j+1}\cdot FLY_{i,j+1},$$

$$(i=1\sim IX,\ j=1\sim IY) \quad \ldots. \quad [3\text{-}2]$$

for three dimensional case,

$$f_{i,j,k}[FLX_{i,j,k}+FRX_{i+1,j,k}+FLY_{i,j,k}+FRY_{i,j+1,k}+FLZ_{i,j,k}+FRZ_{i,j,k+1}$$

$$+AB_{i,j,k}+SD_{i,j,k}]$$

$$= QQ_{i,j,k}+f_{i-1,j,k}\cdot FRX_{i,j,k}+f_{i+1,j,k}\cdot FLX_{i+1,j,k}+f_{i,j-1,k}\cdot FRY_{i,j,k}$$

$$+f_{i,j+1,k}\cdot FLY_{i,j+1,k}+f_{i,j,k-1}\cdot FRZ_{i,j,k}+f_{i,j,k+1}\cdot FLZ_{i,j,k+1},$$

$$(i=1\sim IX,\ j=1\sim IY,\ k=1\sim IZ), \quad [3\text{-}3]$$

where IX, IY and IZ indicate the numbers of coarse meshes in the X—, Y— and Z— directions respectively.

In the case of one dimensional geometry, equation [3-1] is directly solved for the rebalancing factor f_i (i=1,2, ... IX) without iteration process, provided that the value of QQ_i for all cells are given. However, since the QQ_i contains also f_i as shown in expression [1-2], the iteration process is requiredwith the initial value f_i=1 for QQ_i for all i. The $f_i^{(0)}$'s obtained from the first iteration are multiplied to FLX_i, FRX_{i+1}, AB_i, SD_i and FS_i (see Fig. 1) and $f_i^{(1)}$'s are then obtained by solving equation [3-1] again with the source $QQ_i' = f_i^{(0)} \cdot FS_i / k'_{eff}$ ($k'_{eff} = \Sigma_i f_i^{(0)} \cdot FS_i / \Sigma_i QQ_i$) and new coefficients, $FRX_i' = f_{i-1}^{(0)} \cdot FRX_i$ etc. This process is repeated until $f_i^{(n)}$ converges within certain criterion. In two or three dimensional case, it is necessary to solve [3-2] or [3-3] by using an iteration method for the rebalancing factor $f_{i,j}^{(0)}$ (or $f_{i,j,k}^{(0)}$) even if the source $QQ_{i,j}$ (or $Q_{i,j,k}$) are given at the beginning. After obtained the converged value of $f_{i,j}^{(0)}$ (or $f_{i,j,k}^{(0)}$), equation [3-2] (or [3-3]) is modified as in one dimensional case and then the process is repeated to get the next converged value $f_{i,j}^{(1)}$ (or $f_{i,j,k}^{(1)}$).

3.1 One dimensional case

For simplicity, we represent equation [3-1] by the matrix form;

$$A \cdot \vec{f} = \vec{q} \qquad [3\text{-}4]$$

$$A = \begin{pmatrix} a_1 & -b_1 & & & & \\ -c_2 & a_2 & -b_2 & & 0 & \\ & -c_3 & a_3 & -b_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ 0 & & & & -c_{IX} & a_{IX} \end{pmatrix}, \quad \vec{f} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{IX} \end{pmatrix}, \quad \vec{q} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_{IX} \end{pmatrix},$$

where
$$\left.\begin{aligned} a_i &= FLX_i + FRX_{i+1} + AB_i + SD_i \quad (i=1\sim IX), \\ b_i &= FLX_{i+1} \quad (i=1\sim IX-1), \\ c_i &= FRX_i \quad (i=2\sim IX), \\ q_i &= QQ_i \quad (i=1\sim IX). \end{aligned}\right\} \qquad [3\text{-}5]$$

The above matrix equation is solved by the Gauss elimination method. As A is thidiagonal, its algorithm of the forward elimination and backward substitution becomes very simple as follows:

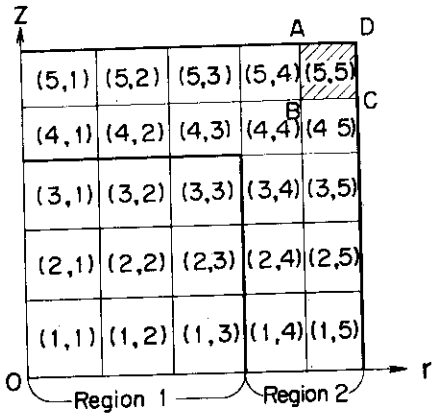


Fig. 4 An example for naming two dimensional cells

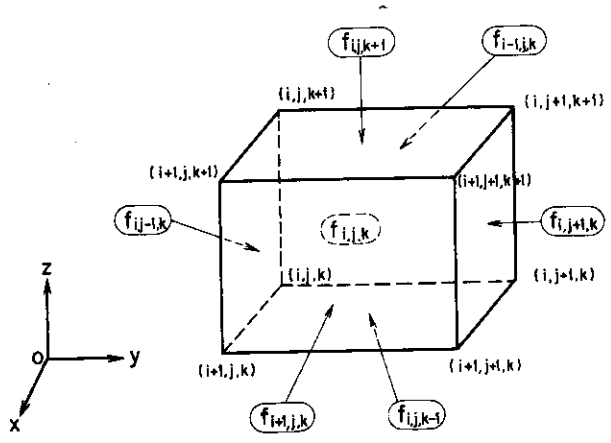


Fig. 5 Three dimensional cell

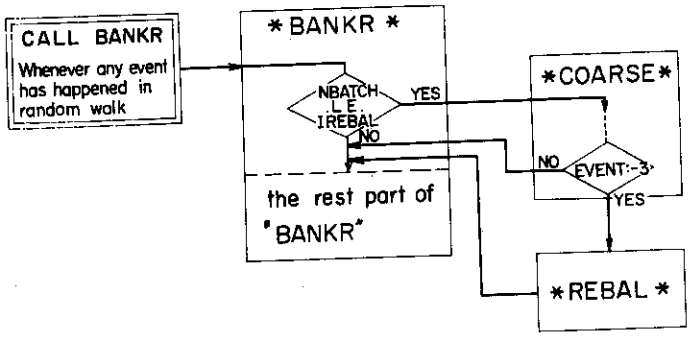


Fig. 6 Modification for the subroutine "BANKR"

forward elimination:

$$\left.\begin{aligned} T_i &= a_i - C_i H_{i-1}, \quad (H_o \equiv 0), \\ G_i &= (q_i + C_i G_{i-1})/T_i, \quad (G_o \equiv 0), \\ H_i &= b_i/T_i, \quad (i=1\sim IX) \end{aligned}\right\} \qquad [3\text{-}6]$$

backward substitution:

$$\left.\begin{aligned} f_{IX} &= G_{IX}, \\ f_i &= G_i + H_i f_{i+1}, \quad (i=IX-1,\sim 1) \end{aligned}\right\} \qquad [3\text{-}7]$$

The rebalancing factors $f_i^{(0)}$'s $(i=1\sim IX)$, obtained from substituting G_i and H_i of [3-6] into [3-7], are multiplied to the coefficients a_i, b_{i-1} and c_{i+1} and new factors $f_i^{(1)}$'s $(i=1\sim IX)$ are recalculated from the balance equation [3-4] with the modified q_i. In general, provided that $f_i^{(n)}$ indicates the rebalancing factor of a cell i in the n-th iteration stage, the iteration formula for $f_i^{(n)}$ is written as follows:

$$A^{(n)} \cdot \vec{f}^{(n)} = \vec{q}^{(n-1)}, \qquad \ldots\ldots \qquad [3\text{-}8]$$

where

$$A^{(n)} = A^{(n-1)} \cdot F^{(n-1)}, \quad F^{(n-1)} = \begin{pmatrix} f_1^{(n-1)} & & & 0 \\ & f_2^{(n-1)} & & \\ & & \ddots & \\ 0 & & & f_{IX}^{(n-1)} \end{pmatrix}.$$

3.2 Two dimensional case

The equation [3-2] is expressed as the matrix equation:

$$\begin{pmatrix} A_1, & -D_1 & & & \\ -E_2, & A_2, & -D_2 & & 0 \\ & -E_3, & A_3, & -D_3, & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & -D_{IY-1} \\ 0 & & & -E_{IY} & A_{IY} \end{pmatrix} \begin{pmatrix} \vec{f}_1 \\ \vec{f}_2 \\ \vec{f}_3 \\ \vdots \\ \vec{f}_{IY} \end{pmatrix} = \begin{pmatrix} \vec{q}_1 \\ \vec{q}_2 \\ \vec{q}_3 \\ \vdots \\ \vec{q}_{IY} \end{pmatrix} \quad \ldots \qquad [3\text{-}9]$$

where A_j, D_j and E_j are IX-dimensional square, tridiagonal matrices, and $\vec{q}_j$ and $\vec{f}_j$ are IX-dimensional vectors defined as follows:

$$A_j = \begin{bmatrix} a_{1,j} & -b_{1,j} & & & \\ -c_{2,j} & a_{2,j} & -b_{2,j} & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -b_{IX-1,j} \\ 0 & & & -c_{IX,j} & a_{IX,j} \end{bmatrix}, \quad (j=1\sim IY),$$

$$a_{i,j} = FLX_{i,j}+FRX_{i+1,j}+FLY_{i,j}+FRY_{i,j+1}+AB_{i,j}+SD_{i,j},$$

$$(i=1\sim IX,\ j=1\sim IY),$$

$$b_{i,j} = FLX_{i+1,j} \qquad (i=1\sim IX-1,\ j=1\sim IY),$$

$$c_{i,j} = FRX_{i,j}, \qquad (i=2\sim IX,\ j=1\sim IY),$$

$$D_j = \begin{bmatrix} d_{1j} & & & 0 \\ & d_{2j} & & \\ & & \ddots & \\ 0 & & & d_{IX,j} \end{bmatrix}, \quad d_{i,j} = FLY_{i,j+1}, \quad (i=1\sim IX,\ j=1\sim IY-1),$$

$$E_j = \begin{bmatrix} e_{1,j} & & & 0 \\ & e_{2,j} & & \\ & & \ddots & \\ 0 & & & e_{IX,j} \end{bmatrix}, \quad e_{i,j} = FRY_{i,j}, \quad (i=1\sim IX,\ j=2\sim IY),$$

$$\vec{q}_j = (q_{1j},\ q_{2j} \ \ldots\ldots q_{IX,j})^T, \quad (j=1\sim IY),$$

$$q_{i,j} = QQ_{i,j}, \qquad (i=1\sim IX,\ j=1\sim IY),$$

$$\vec{f}_j = (f_{1,j},\ f_{2,j} \ \ldots\ldots\ f_{IX,j})^T, \quad (j=1\sim IY)\ .$$

Putting $E_1=D_{IT}=0$ at the boundaries, equation [3-9] is equivalent to a set of IY equations:

$$-E_j\vec{f}_{j-1}+A_j\vec{f}_j-D_j\vec{f}_{j+1} = \vec{q}_j, \quad (j=1\sim IY). \qquad \ldots\ldots \qquad [3\text{-}10]$$

Before calculating the n-th rebalancing factors $\vec{f}_j^{(n)}$ $(j=1\sim IY)$ from the $\vec{f}_j^{(n-1)}$ $(j=1\sim IY)$, it is necessary to have a convergence process for the iteration scheme of $\vec{f}_j^{(n-1)}$ according to formula [3-10].

The m-th iteration values $\vec{f}_j^{(n,m)}$'s at the n-th stage are computed from the following equation:

$$A_j^{(n)} \cdot \vec{f}_j^{(n,m)} = \vec{q}_j^{(n-1)} + E_j^{(n)} \cdot \vec{f}_{j-1}^{(n,m)} + D_j^{(n)} \cdot \vec{f}_{j+1}^{(n,m-1)} \equiv \vec{q}_j^{(n-1,m-1)} \quad (j=1\sim IY), \qquad \cdots\cdots \quad [3\text{-}11]$$

which are solved similarly to the one dimensional case by the forward and backward algorithm because $A_j^{(n)}$'s are also tri-diagonal matrices. The $\vec{f}_{j-1}^{(n,m)}$ on the right hand side of equation [3-11] are substituted for $\vec{f}_{j-1}^{(n,m-1)}$ to utilize the newest information in the iteration procedure.

The initial values $f_j^{(o,o)}$'s are assumed to be 1 and starting values $f_j^{(n,o)}$'s at the n-th stage are the newest values at the previous stage $f_j^{(n-1,newest)}$'s $(j=2\sim IY)$. After converged the iteration procedure, the square matrices $A_j^{(n)}$, $E_j^{(n)}$ and $D_j^{(n)}$, and $\vec{q}_j^{(n-1)}$ are modified according to the next equations as in one dimensional case:

$$\left.\begin{aligned} A_j^{(n)} &= A_j^{(n-1)} \cdot F_j^{(n-1)}, \\ E_j^{(n)} &= E_j^{(n-1)} \cdot F_j^{(n-1)}, \\ D_j^{(n)} &= D_j^{(n-1)} \cdot F_j^{(n-1)}, \\ q_{i,j}^{(n)} &= FS_{i,j}^{(n)} / k_{eff}^{(n)}, \quad (i=1\sim IX) \end{aligned}\right\} \qquad \cdots\cdots \quad [3\text{-}12]$$

where

$$F_j^{(n-1)} = \begin{bmatrix} f_{1j}^{(n-1)} & & & 0 \\ & f_{2j}^{(n-1)} & & \\ & & \ddots & \\ 0 & & & f_{IX,j}^{(n-1)} \end{bmatrix}, \quad (j=1\sim IY),$$

$$FS_{i,j}^{(n)} = f_{i,j}^{(n-1)} \cdot FS_{i,j}^{(n-1)}, \quad k_{eff}^{(n)} = \sum_{i,j} FS_{ij}^{(n)} / \sum_{i,j} q_{i,j}^{(o)},$$

in which $f_{i,j}^{(n-1)}$'s stand for the converged values.

3.3 Three dimensional case

We represent also equation [3-3] as a matrix form:

$$\begin{pmatrix} S_1, -T_1 & & & 0 \\ U_2, & S_2, -T_2 & & \\ & -U_3, & S_3, -T_3 & \\ & & \ddots & \ddots \\ & & & -T_{IZ-1} \\ 0 & & & -U_{IZ}, S_{IZ} \end{pmatrix} \begin{pmatrix} \vec{f}_1 \\ \vec{f}_2 \\ \vec{f}_3 \\ \vdots \\ \vec{f}_{IZ} \end{pmatrix} = \begin{pmatrix} \vec{q}_1 \\ \vec{q}_2 \\ \vec{q}_3 \\ \vdots \\ \vec{q}_{IZ} \end{pmatrix} . \qquad [3\text{-}13]$$

Here S_k's ($k=1\sim IZ$) are ($IX\times IY$)-dimensional tridiagonal matrices, T_k and U_k ($k=1\sim IZ$) are ($IX\times IY$) diagonal matrices, and $\vec{f}_k$ and $\vec{q}_k$ are ($IX\times IY$) dimensional vectors defined respectively as follows:

$$S_k = \begin{pmatrix} A_{1,k} & -D_{1,k} & & 0 \\ -E_{2,k} & A_{2,k} & -D_{2,k} & \\ & \ddots & \ddots & \ddots \\ & & \ddots & -D_{IY-1,k} \\ 0 & & -E_{IY,k} & A_{IY,k} \end{pmatrix}, (k=1\sim IZ), \quad .... \qquad [3\text{-}14]$$

where

$$A_{j,k} = \begin{pmatrix} a_{1,j,k} & -b_{1,j,k} & & 0 \\ -c_{2,j,k} & a_{2,j,k} & -b_{2,j,k} & \\ & \ddots & \ddots & \ddots \\ & & \ddots & -b_{IX-1,j,k} \\ 0 & & -c_{IX,j,k} & a_{IX,j,k} \end{pmatrix}, (j=1\sim IY) \quad .... \qquad [3\text{-}15]$$

$$D_{j,k} = \begin{pmatrix} d_{1,j,k} & & & 0 \\ & d_{2,j,k} & & \\ & & \ddots & \\ 0 & & & d_{IX,j,k} \end{pmatrix}, (j=1\sim IY) \quad .... \qquad [3\text{-}16]$$

$$E_{j,k} = \begin{pmatrix} e_{1,j,k} & & 0 \\ & e_{2,j,k} & \\ & \ddots & \\ 0 & & e_{IX,j,k} \end{pmatrix}, (j=1\sim IY). \quad ... \qquad [3\text{-}17]$$

The elements of these matrices are given by

$$a_{i,j,k} = FLX_{i,j,k}+FRX_{i+1,j,k}+FLY_{i,j,k}+FRY_{i,j+1,k}+FLZ_{i,j,k}+FRZ_{i,j,k+1}$$
$$+AB_{i,j,k}+SD_{i,j,k},$$

$$b_{i,j,k} = FLX_{i+1,j,k},$$

$$c_{i,j,k} = FRX_{i,j,k},$$

$$d_{i,j,k} = FLY_{i,j+1,k},$$

$$e_{i,j,k} = FRY_{i,j,k}.$$

The matrices T_k and U_k are written respectively in the forms:

$$T_k = \begin{bmatrix} T'_{1,k} & & 0 \\ & T'_{2,k} & \\ & & \ddots \\ 0 & & T'_{IY,k} \end{bmatrix}, \quad (k=1\sim IZ), \qquad [3\text{-}18]$$

$$U_k = \begin{bmatrix} U'_{1,k} & & 0 \\ & U'_{2,k} & \\ & & \ddots \\ 0 & & U'_{IY,k} \end{bmatrix}, \quad (k=1\sim IZ), \qquad [3\text{-}19]$$

where $T'_{j,k}$ and $U'_{j,k}$ $(j=1\sim IY)$, elements of which are represented respectively as $t_{i,j,k} = FLZ_{i,j,k+1}$ and $U_{i,j,k} = FRZ_{i,j,k}$ are IX-dimensional diagonal matrices:

$$T'_{j,k} = \begin{bmatrix} t_{1,j,k} & & 0 \\ & t_{2,j,k} & \\ & & \ddots \\ 0 & & t_{IX,j,k} \end{bmatrix}, \quad (j=1\sim IY,\ k=1\sim IZ)$$

$$U'_{j,k} = \begin{bmatrix} u_{1,j,k} & & 0 \\ & u_{2,j,k} & \\ & & \ddots \\ 0 & & u_{IX,j,k} \end{bmatrix}, \quad (j=1\sim IY,\ k=1\sim IZ)$$

Furthermore we can decompose the (IX·IY)-dimensional vectors $\vec{f}_k$ and $\vec{q}_k$ as follows:

$$\left.\begin{aligned}\vec{f}_k &= (\vec{f}'_{1,k}, \vec{f}'_{2,k}, \cdots\cdots, \vec{f}'_{IY,k})^T \\ \vec{q}_k &= (\vec{q}'_{1,k}, \vec{q}'_{2,k}, \cdots\cdots, \vec{q}'_{IY,k})^T\end{aligned}\right\} , \quad \cdots\cdots \quad [3\text{-}20]$$

where

$$\left.\begin{aligned}\vec{f}'_{j,k} &= (f_{1,j,k}, f_{2,j,k}, \cdots\cdots, f_{IX,j,k})^T \\ \vec{q}'_{j,k} &= (q_{1,j,k}, q_{2,j,k}, \cdots\cdots, q_{IX,j,k})^T\end{aligned}\right\} , \quad (j=1\sim IY,\ k=1\sim IZ),$$

in which $q_{i,j,k} = QQ_{i,j,k}$ and $f_{i,j,k}$ indicates the rebalancing factor for a cell [i,j,k].

Now putting formally $U_1 = T_{IZ} = 0$ in equation [3-13], a set of IZ equations equivalent to [3-13] is obtained:

$$-U_k\vec{f}_{k-1} + S_k\vec{f}_k - T_k\vec{f}_{k+1} = \vec{q}_k , \quad (d=1\sim IZ). \qquad [3\text{-}21]$$

Setting $E_{1,k} = D_{IY,k} = 0$ in [3-14] and $U'_{j,1} = T'_{j,IZ} = 0$, which is derived from $U_1 = T_{IZ} = 0$ using equation [3-18] and [3-19], [3-21] is decomposed to

$$-U'_{j,k}\vec{f}'_{j,k-1} - E_{j,k}\vec{f}'_{j-1,k} + A_{j,k}\vec{f}'_{j,k} - D_{j,k}\vec{f}'_{j+1,k} - T'_{j,k}\vec{f}'_{j,k+1} = \vec{q}'_{j,k}$$

$$(j=1\sim IY,\ k=1\sim IZ) \quad \cdots \quad [3\text{-}22]$$

Equation [3-22] leads now to the following iteration scheme as in two dimensional case and it can be solved by the algorithm described already because $A^{(n)}_{j,k}$'s are tridiagonal matrices.

$$A^{(n)}_{j,k}\vec{f}'^{(n,m)}_{j,k} = \vec{q}^{(n-1,m-1)}_{j,k}$$

$$\vec{q}^{(n-1,m-1)} = \vec{q}'^{(n-1)}_{j,k} + U'^{(n)}_{j,k}\cdot\vec{f}'^{(n,m)}_{j,k-1} + E^{(n)}_{j,k}\cdot\vec{f}'^{(n,m)}_{j-1,k} + D^{(n)}_{j,k}\cdot\vec{f}'^{(n,m-1)}_{j+1,k} + T'^{(n)}_{j,k}\cdot\vec{f}'^{(n,m-1)}_{j,k+1} ,$$

$$\cdots\cdots \quad [3\text{-}23]$$

where

$$A^{(n)}_{j,k} = A^{(n-1)}_{j,k}\cdot F^{(n-1)}_{j,k} ,$$

$$U'^{(n)}_{j,k} = U'^{(n-1)}_{j,k}\cdot F^{(n-1)}_{j,k} ,$$

$$E^{(n)}_{j,k} = E^{(n-1)}_{j,k}\cdot F^{(n-1)}_{j,k} ,$$

$$D^{(n)}_{j,k} = D^{(n-1)}_{j,k} \cdot F^{(n-1)}_{j,k} ,$$

$$T'^{(n)}_{j,k} = T'^{(n-1)}_{j,k} \cdot F^{(n-1)}_{j,k} ,$$

$$F^{(n-1)}_{j,k} = \begin{bmatrix} f^{(n-1)}_{1,j,k} & & & 0 \\ & f^{(n-1)}_{2,j,k} & & \\ & & \ddots & \\ 0 & & & f^{(n-1)}_{IX,j,k} \end{bmatrix} , \quad (j=1\sim IY,\ k=1\sim IZ).$$

3.4 Boundary conditions, convergence criteria and remarks

The subroutine "REBAL" has two options for the boundary condition of a reactor, vacuum and reflective conditions. We describe here the treatment of the boundary condition in one dimensional geometry only, because the extention can easily be performed to two or three dimensional case.

(i) For the vacuum boundary condition at both ends of a reactor (I=1 and I=IX+1),

$$FRX_1 = c_1 = 0 , \quad FLX_{IX+1} = b_{IX} = 0$$

Hence, it is not necessary to alter equation [3-5].

(ii) For the reflective boundary condition at the left surface (or center) of a reactor, putting $f_0 = f_1$ and $FRX_1 = FLX_1$ in equation [3-1] with i=1, we get

$$(FRX_2 + AB_1 + SD_1)\cdot f_1 - FLX_2 \cdot f_2 = QQ_1 .$$

So the first term on the right hand side of a_i in [3-5] vanishes at i=1, namely

$$a_1 = FR_2 + AB_1 + SD_1 .$$

Similarly, for the reflective boundary condition at the right surface,

$$a_{IX} = FLX_{IX} + AB_{IX} + SD_{IX} .$$

In the "REBAL", the convergence of the rebalance factor is judged from the following criterion:

$$\left| 1.0 - \frac{k_{eff}^{(n)}}{k_{eff}^{(n-1)}} \right| < \varepsilon_3$$

This means that the relative change in the total number of fission neutrons produced in the system becomes less than ε_3 after the coarse mesh scaling is applied n times to it. In two or three dimensional case, in addition, it is judged if the rebalancing factors converge in the inner iterative loop. For example, the criterion for two dimensional rebalancing factor $f_L^{(n,m)}$ is given as

$$\max_L \left| 1.0 - \frac{f_L^{(n,m)}}{f_L^{(n,m-1)}} \right| < \varepsilon_0 \quad .$$

If the computed rebalancing factor becomes negative, if the absolute value of a denominator in the forward algorithm of Gauss method [3-6] becomes extremely small, or if the standard deviation of effective multiplication factor k_{eff} is very large relative to the difference between the rebalancing factor and 1, we use one rebalancing factor f for the whole system, given by

$$f = \frac{\sum_L (QQ)_L}{NL + \sum_L (AB)_L} \quad .$$

The NL represents the net leakage from the whole system and is given for each dimensional case as follows:

one dimensional case;

$$NL = FLX_1 + FRX_{IX+1} - [FRX_1 + FLX_{IX+1}],$$

two dimensional case;

$$NL = \sum_j (FLX_{1,j} + FRX_{IX+1,j}) + \sum_i (FLY_{i,1} + FRY_{i,IY+1})$$

$$-[\sum_j (FRX_{1,j} + FLX_{IX+1,j}) + \sum_i (FRY_{i,1} + FLY_{i,IY+1})],$$

three dimensional case;

$$NL = \sum_{j,k} (FLX_{1,j,k}+FRX_{IX+1,j,k}) + \sum_{i,k} (FLY_{i,1,k}+FRY_{i,IY+1,k}) + \sum_{i,j} (FLZ_{i,j,1}+FRZ_{i,j,IZ+1})$$

$$-[\sum_{j,k} (FRX_{1,j,k}+FLX_{IX+1,j,k}) + \sum_{i,k} (FRY_{i,1,k}+FLY_{i,IY+1,k})$$

$$+ \sum_{i,j} (FRZ_{i,j,1}+FLZ_{i,j,IZ+1})],$$

where the expressions shown in brackets are for the reflective boundary condition.

4. Interface between "COARSE" and "REBAL"

(i) One dimensional case;

Argument variable	Dimension	Content
FS	IR	$(FS)_\ell$
FR	IRP	$(FRX)_\ell$
FL	IRP	$(FLX)_\ell$
AB	IR	$(AB)_\ell$
SD	IR	$(SD)_\ell$
QQS		SWATE
VAR		Standard deviation of k_{eff}
EPS1		Measure for convergence
EPS2		$(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4)$
EPS3		
EPS4		
IR		Total number of cells
IRP		IR+1
NBATCH		Batch number
NBDL(*)		Boundary condition at left edge
NBDR(*)		Boundary condition at right edge
FKT		$[k_{eff}]^{old}$ from MORSE
IEND		Flag for normal job end

(ii) Two dimensional case

Argument variable	Dimension	Content
FS	(IRX, IRY)	$FS(\ell,\ell')$

$$NL = \sum_{j,k} (FLX_{1,j,k}+FRX_{IX+1,j,k}) + \sum_{i,k} (FLY_{i,1,k}+FRY_{i,IY+1,k}) + \sum_{i,j} (FLZ_{i,j,1}+FRZ_{i,j,IZ+1})$$

$$-[\sum_{j,k} (FRX_{1,j,k}+FLX_{IX+1,j,k}) + \sum_{i,k} (FRY_{i,1,k}+FLY_{i,IY+1,k})$$

$$+ \sum_{i,j} (FRZ_{i,j,1}+FLZ_{i,j,IZ+1})],$$

where the expressions shown in brackets are for the reflective boundary condition.

4. Interface between "COARSE" and "REBAL"

(i) One dimensional case;

Argument variable	Dimension	Content
FS	IR	$(FS)_\ell$
FR	IRP	$(FRX)_\ell$
FL	IRP	$(FLX)_\ell$
AB	IR	$(AB)_\ell$
SD	IR	$(SD)_\ell$
QQS		SWATE
VAR		Standard deviation of k_{eff}
EPS1		Measure for convergence
EPS2		$(\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4)$
EPS3		
EPS4		
IR		Total number of cells
IRP		IR+1
NBATCH		Batch number
NBDL(*)		Boundary condition at left edge
NBDR(*)		Boundary condition at right edge
FKT		$[k_{eff}]^{old}$ from MORSE
IEND		Flag for normal job end

(ii) Two dimensional case

Argument variable	Dimension	Content
FS	(IRX, IRY)	$FS(\ell,\ell')$

CE	(IRXP, IRY)	FRX(ℓ,ℓ')
CW	(IRXP, IRY)	FLX(ℓ,ℓ')
CN	(IRX, IRYP)	FRY(ℓ,ℓ')
CS	(IRX, IRYP)	FLY(ℓ,ℓ')
AB	(IRX, IRY)	AB (ℓ,ℓ')
SD	(IRX, IRY)	SD (ℓ,ℓ')
QQS		SWATE
VAR		Standard deviation of k_{eff}
EPS0		Measure for convergence (ε_0, ε_1, ε_2, ε_3, ε_4)
EPS1		
EPS2		
EPS3		
EPS4		
IRX		Number of cells in r-direction
IRY		Number of cells in z-direction
IRXP		IRX+1
IRYP		IRY+1
NBATCH		Batch number
NBDXL(*)		Boundary condition at left edge
NBDXR(*)		Boundary condition at right edge
NBDYL(*)		Boundary condition at top edge
NBDYR(*)		Boundary condition at bottom edge
FKT		$k_{eff}^{(old)}$ from MORSE
IEND		Flag for normal job end
RAMBD		Converged value for $\sum_{\ell,\ell'}$ FS(ℓ,ℓ')

(*) for indeces for boundary conditions;
= 0 : vacuum boundary
= 1 : reflective boundary.

5. References

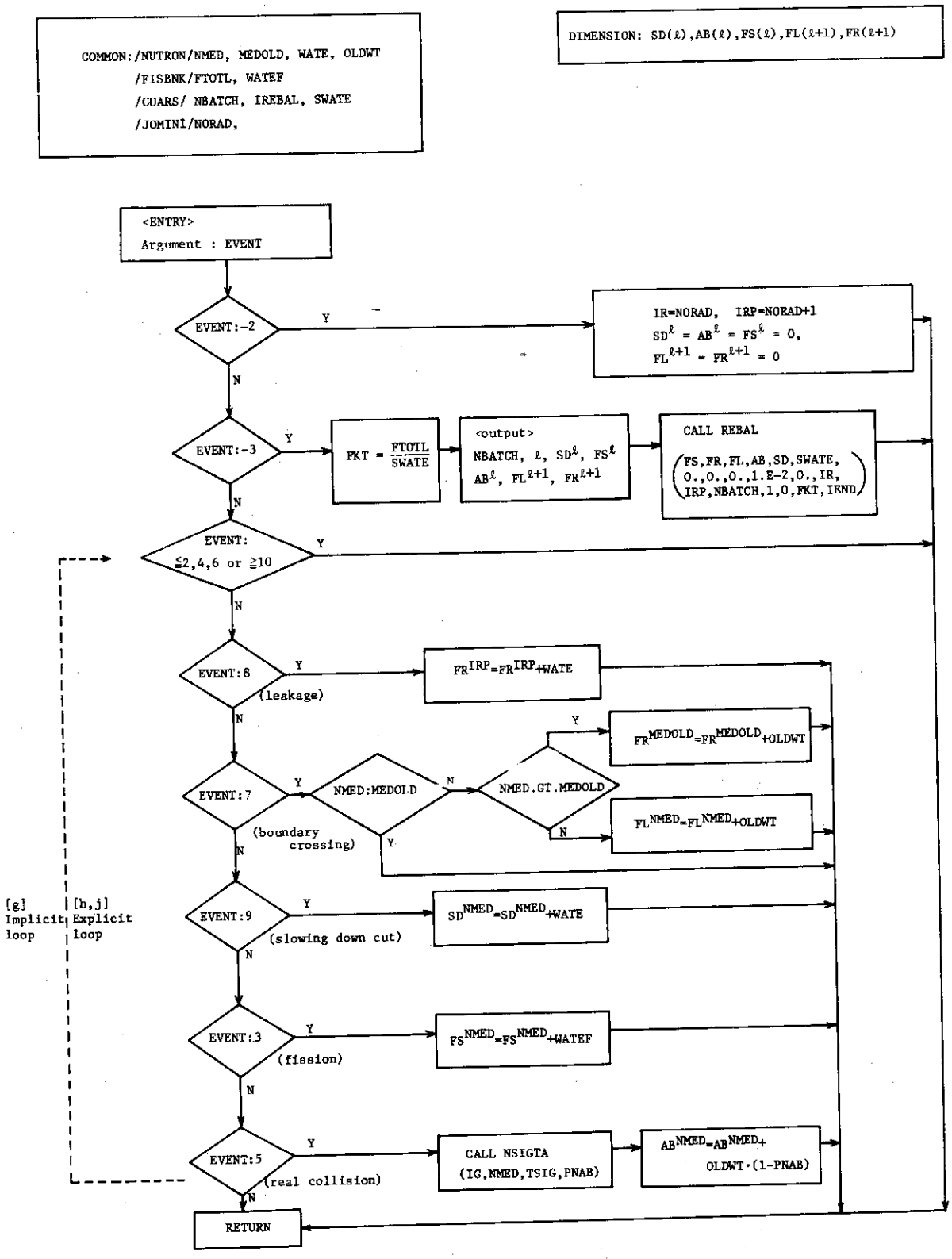
1) T. ASAOKA et al., "Coarse-Mesh Rebalancing Acceleration for Eigenvalue Problems," in Proc. NEACRP Meeting of a Monte Carlo Study Group, ANL-75-2 (NEACRP-L-118), Argonne National Laboratory (1975).

2) T. ASAOKA et al., "Application of Coarse-Mesh Rebalance Acceleration to Monte Carlo Eigenvalue Problems," to be submitted to Nucl. Sci. Eng. (1975).

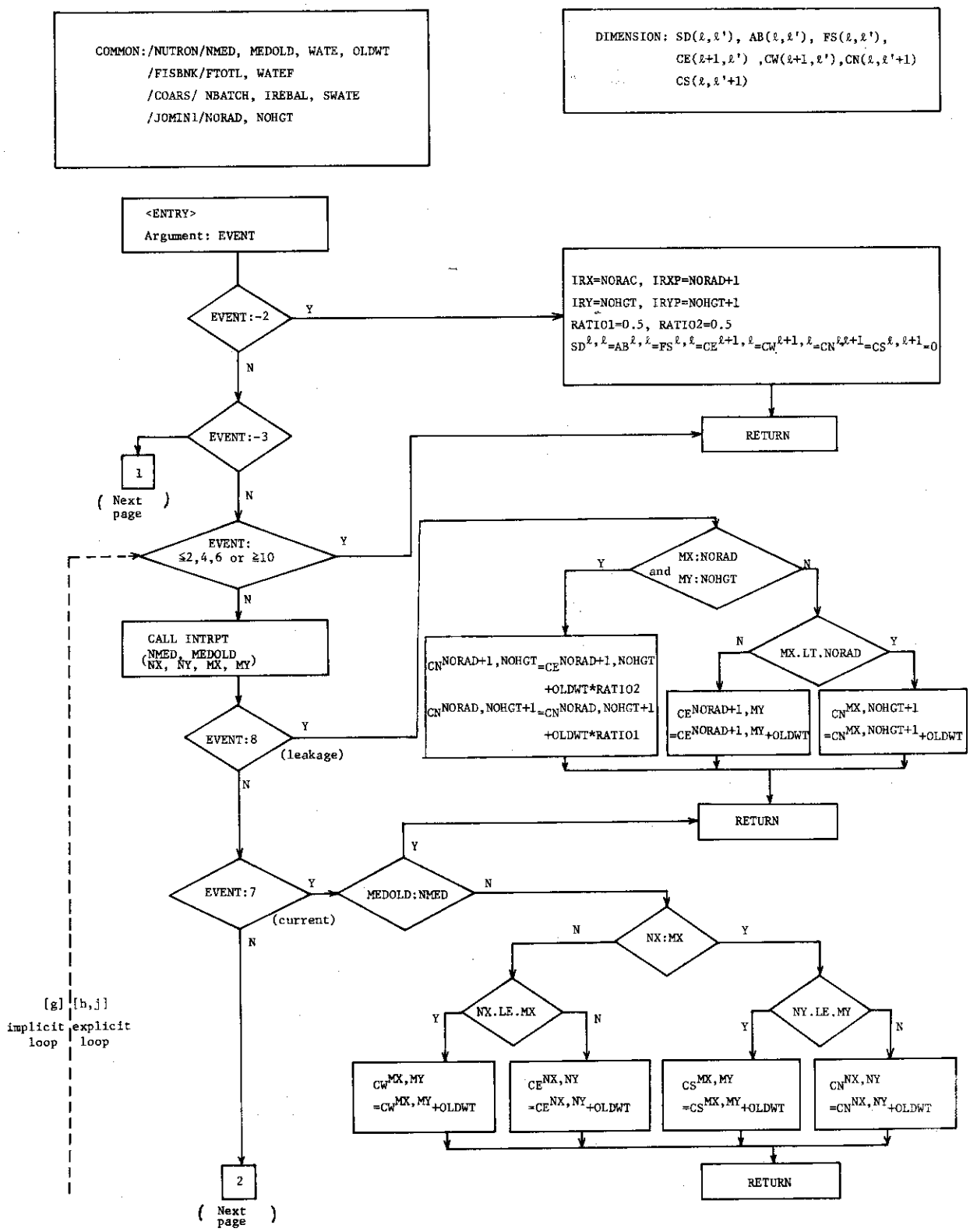
3) E. A. STRAKER et al., "The MORSE Code-A Multigroup Neutron and Gamma-Ray Monte Carlo Transport Code," ORNL-4585, Oak Ridge National Laboratory (1970).

4) V. R. CAIN, "SAMBO, A Collision Analysis Package for Monte Carlo Doses," ORNL-TM-3203, Oak Ridge National Laboratory (1970).

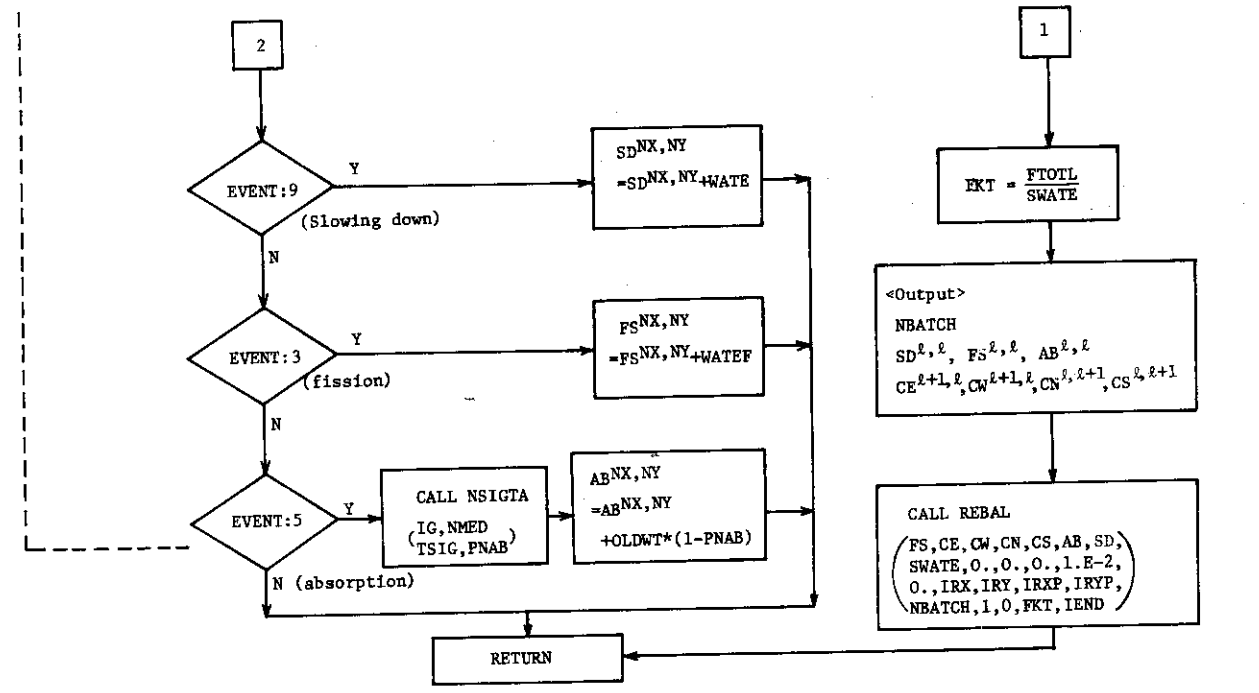
APPENDIX I Flow chart of one dimensional "COARSE"



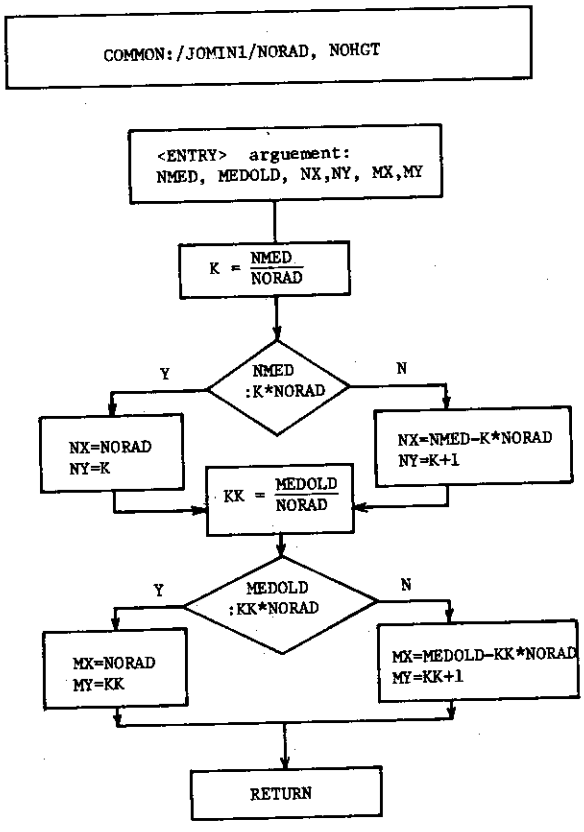
APPENDIX II Flow chart of two dimensional "COARS2"



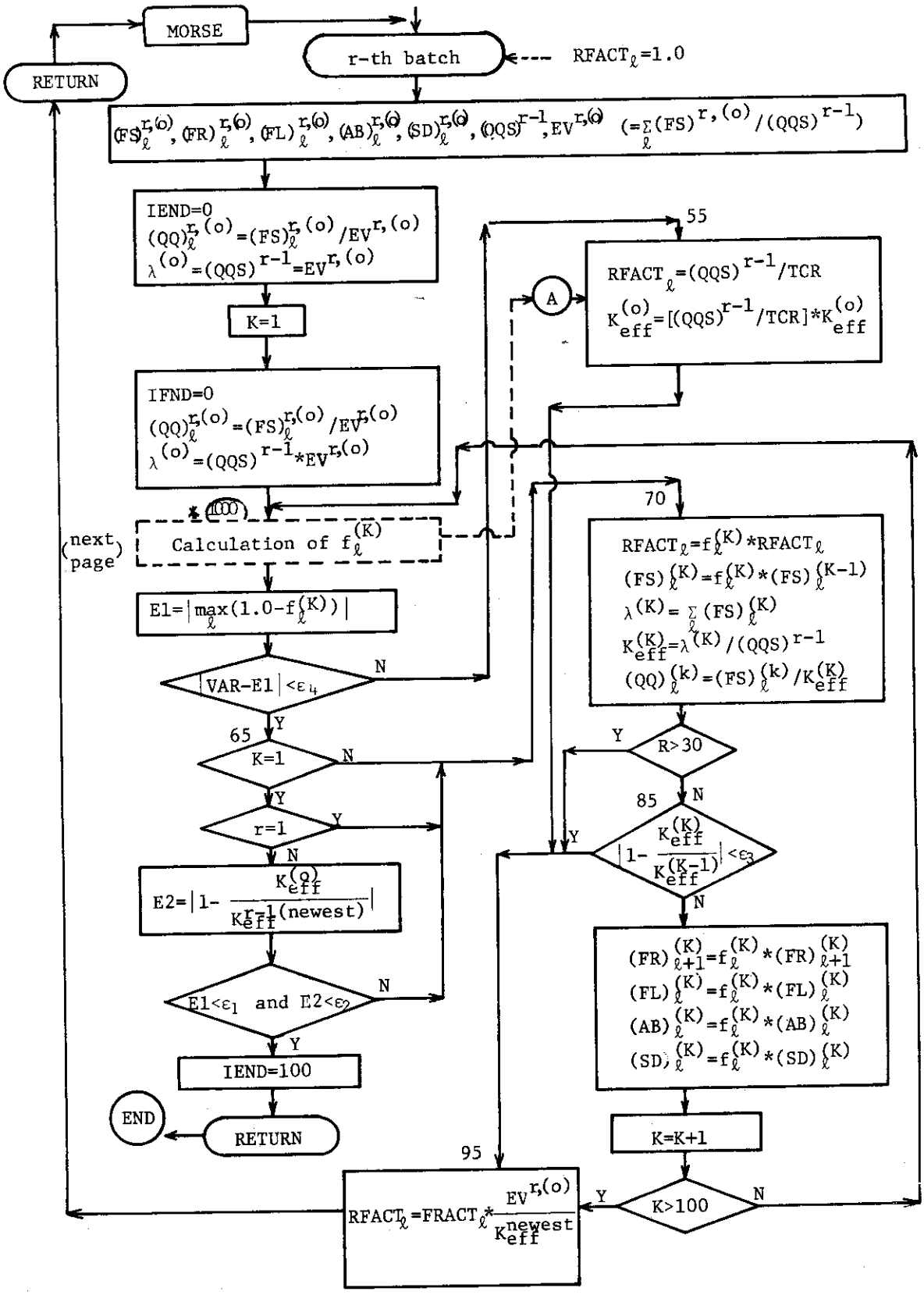
APPENDIX II Flow chart of two-dimensional "COARS2" (continued)



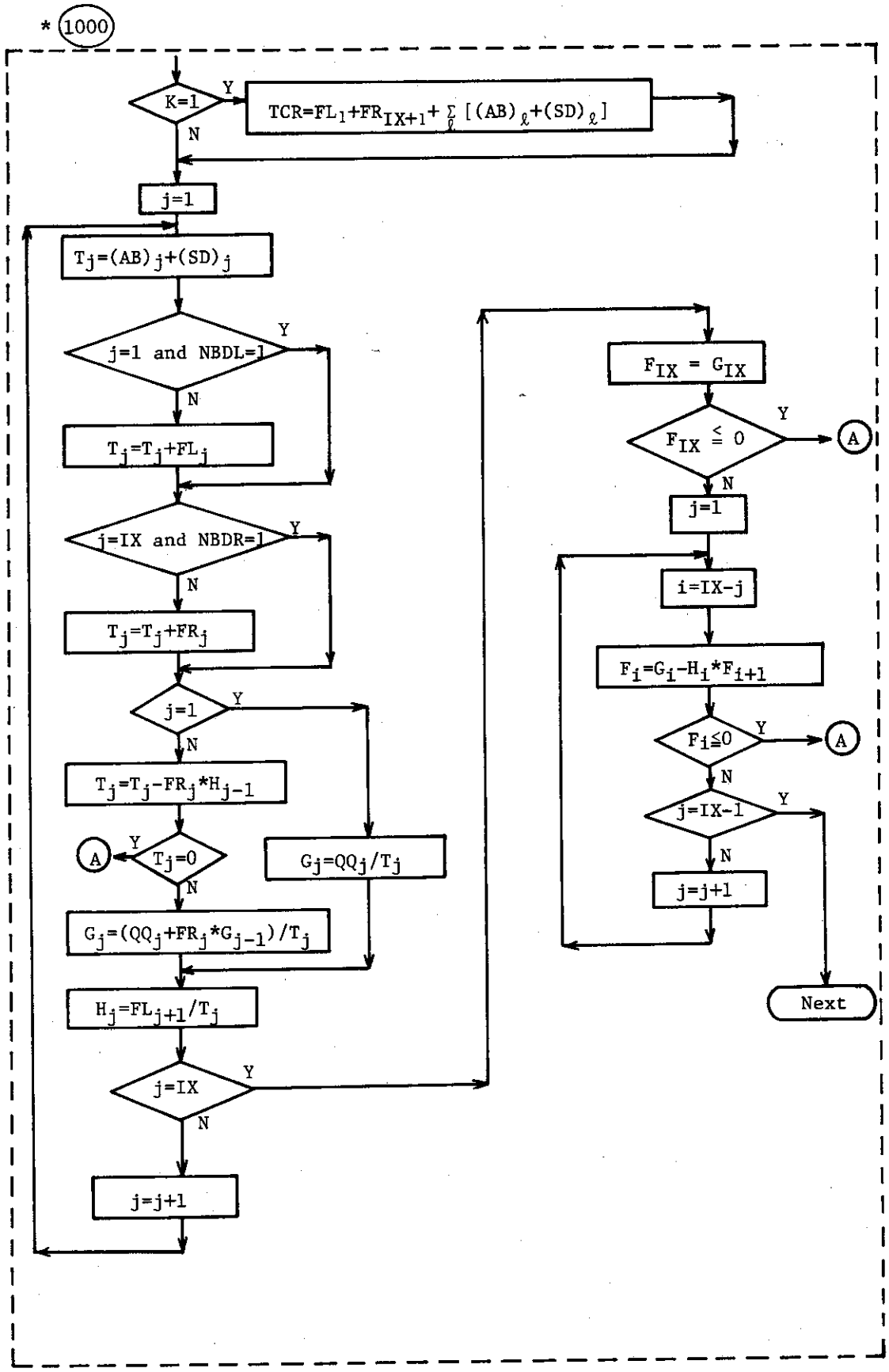
Subroutine INTPRT



APPENDIX III. Flow chart of one dimensional "REBAL"



APPENDIX III Flow chart of one dimensional "REBAL"
(calculation of the scaling factor $f_{\ell}^{(K)}$)



APPENDIX IV Program list of "COARSE", "COARS3", "REBAL", and "REBAL3"

```
      SUBROUTINE COARSE(EVENT)
      INTEGER EVENT
      COMMON/APOLLO/DUM(30),IR
      COMMON /NUTRON/ NAME,NAMEX,IG,IGO,NMED,MEDOLD,NREG,U,V,W,UOLD,VOLD
     1 ,WOLD,X,Y,Z,XOLD,YOLD,ZOLD,WATE,OLDWT,WTBC,BLZNT,BLZON,AGE,OLDAGE
      COMMON /FISBNK/ MFISTP,NFISBN,NFISH,FTOTL,FWATE,WATEF
C     COMMON/FISBN/FISBNK(252),FTOTL,FWATE,MFISTP,NFISH,FPT,WATEF
      COMMON /COARS/ NBATCH,IREBAL,SWATE
      DIMENSION       SD(10),AB(10),FS(10),CUR(10,10),FL(11),FR(11)
      IF(EVENT.EQ.-1.OR.EVENT.EQ.-2.OR.EVENT.EQ.-3)WRITE(6,1)
     1IR,EVENT,IRP
    1 FORMAT(1HO,'**  IR =',I5,10X'EVENT =',I5,10X'IRP =',I5)
      IF(EVENT.NE.-2) GO TO 100
      IRP=IR+1
      TCR=0.
      DO 10 L=1,10
      SD(L)=0.
      AB(L)=0.
      FS(L)=0.
      FL(L)=0.
   10 FR(L)=0.
      FL(11)=0.
      FR(11)=0.
      DO 20 L=1,IR
      DO 20 LD=1,IR
   20 CUR(L,LD)=0.
      RETURN
  100 CONTINUE
      IF(EVENT.EQ.-3) GO TO 700

C
C     *  COUNTS EVERY EVENTS IN MEDIUM (L)  *
C
      IF(EVENT.LE.2.OR.EVENT.EQ.4.OR.EVENT.GE.10.OR.EVENT.EQ.6) RETURN
      IF(EVENT.NE.8) GO TO 307
      TCR=TCR+WATE
      GO TO 400
  307 IF(EVENT.NE.7) GO TO 309
      IF(MEDOLD.EQ.NMED) GO TO 400
      CUR(MEDOLD,NMED)=CUR(MEDOLD,NMED)+OLDWT
      GO TO 400
  309 IF(EVENT.NE.9) GO TO 303
      SD(NMED)=SD(NMED)+WATE
      GO TO 400
  303 IF(EVENT.NE.3) GO TO 305
      FS(NMED)=FS(NMED)+WATEF
      GO TO 400
  305 IF(EVENT.NE.5) GO TO 400
      CALL NSIGTA(IG,NMED,TSIG,PNAB)
      AB(NMED)=AB(NMED)+OLDWT*(1.-PNAB)
  400 RETURN
C
C
  700 CONTINUE
      DO 510 M=1,IR
      FR(M+1)=0.
      FL(M+1)=0.
      DO 510 L=1,IR
      DO 510 LD=1,IR
      IF(L.LE.M) GO TO 500
      IF(LD.GT.M) GO TO 510
      FL(M+1)=FL(M+1)+CUR(L,LD)
      GO TO 510
  500 IF(LD.LE.M) GO TO 510
      FR(M+1)=FR(M+1)+CUR(L,LD)
  510 CONTINUE
      FR(IRP)=TCR
      FKT=FTOTL/SWATE
C
C
      WRITE(6,620) NBATCH
  620 FORMAT(//10X,21H** BATCH NUMBER **  (,I4,2H) //)
      WRITE(6,600)(L,      FS(L),AB(L),SD(L), L=1,IR)
  600 FORMAT(//10X, 65HMEDIUM NO.                              FS
     1AB             SD/(14X,I2,11X,14X,     E12.5,2X,E12.5,2X,E12.5))
      WRITE(6,610)(L,FL(L),FR(L),L=1,IRP)
  610 FORMAT(//10X, 37HMEDIUM NO.            FL             FR/(14X,I2,11X
     1,E12.5,2X,E12.5))
C
      CALL REBAL(FS,FR,FL,AB,SD,SWATE,0.,0.,0.,1.E-2,0., IR,IRP,NBATCH,
     1 1.0, FKT,IEND,FTOTL)
CCC   1 1.0, FKT,IEND)
      RETURN
      END
```

Appindix IV-2

```
      SUBROUTINE COARS3(EVENT)
C
C     * COARSE ROUTINE FOR TWO DIMENSION GEOMETRY (CYLINDER)
C
      INTEGER EVENT
      DOUBLE PRECISION  RADIUS,HIGHT,HMAX,RMAX,BIG,XDUM,YDUM,ZDUM
      COMMON/APOLLO/DUM(30),IR
      COMMON/NUTRON/ NAME,NAMEX,IG,IGO,NMED,MEDOLD,NREG,U,V,W,UOLD,VOLD,
     1  WOLD,X,Y,Z,XOLD,YOLD,ZOLD,WATE,OLDWT,WTBC,BLZNT,BLZON,AGE,OLDAGE
      COMMON/FIGBNK/NFISTP,NFISBN,NFISH,FTOTL,FWATE,WATEF
      COMMON/COARS/NBATCH,IREBAL,SWATE
      COMMON/JOININ1/RADIUS(20,20),HIGHT(20),HMAX,RMAX,BIG,NORAD,NOHGT,
     1 MED(20,20),IVOID,IRAD,IHGT
      DIMENSION SD(10,10),AB(10,10),FS(10,10),CE(11,10),C*(11,10),CN(10,
     110,11),CS(10,10,11)
C
      IF(EVENT.NE.-2) GO TO 100
      IRX=NORAD
      IRY=NOHGT
      IRXP=IRX+1
      IRYP=IRY+1
      RAT1=RMAX
      DO 11 L=1,NORAD
      LD=L
      IF(RMAX-RADIUS(NOHGT,L)*1.001) 12,12,11
   11 CONTINUE
   12 IF(LD.EQ.1) GO TO 13
      RNEXT=RADIUS(NOHGT,LD-1)
      RAT1=RMAX-RNEXT
   13 RAT8=RMAX
      DO 14 L=1,NORAD
      LD=L
      IF(RMAX-RADIUS(1,L)*1.001) 15,15,14
   14 CONTINUE
   15 IF(LD.EQ.1) GO TO 16
      RNEXT=RADIUS(1,LD-1)
      RAT8=RMAX-RNEXT
   16 RNEXT=DSQRT(RMAX)
      RAT2=2.*RNEXT*(HIGHT(NOHGT)-HIGHT(NOHGT-1))
      RATA=2.*RNEXT*HIGHT(1)
      RATIO1=RAT1/(RAT1+RAT2)
      RATIB1=RAT8/(RAT8+RATA)
      RATIO2=1.-RATIO1
      RATIB2=1.-RATIB1
C
      DO 20 L=1,IRX
      DO 20 LD=1,IRY
      SD(L,LD)=0.
      AB(L,LD)=0.
   20 FS(L,LD)=0.
      DO 30 L=1,IRXP
      DO 30 LD=1,IRY
      CE(L,LD)=0.
   30 C*(L,LD)=0.
      DO 40 L=1,IRX
      DO 40 LD=1,IRYP
      DO 40 I=1,IRX
      CN(L,I,LD)=0.0
   40 CS(L,I,LD)=0.0
      RETURN
  100 CONTINUE
      IF(EVENT.EQ.-3) GO TO 700
```

```
      C
      C     *  COUNTS EVERY EVENTS IN MEDIUM(I,J)
      C
   55       IF(EVENT.LE.2.OR.EVENT.EQ.4.OR.EVENT.EQ.6.OR.EVENT.GE.10) RETURN
   56       IRADN=IRAD
   57       IHGTN=IHGT
   58       XDUM=XOLD
   59       YDUM=YOLD
   60       ZDUM=ZOLD
   61       CALL LOOKZ(XDUM,YDUM,ZDUM)
   62       IF(EVENT.NE.8) GO TO 307
   63       IF(IHGT-NOHGT) 21,22,22
   64    22 IF(RADIUS(IHGT,IRAD)-RMAX*.999) 318,318,328
   65   328 CE(IRAD+1,IHGT)=CE(IRAD+1,IHGT)     +OLDWT*RATIO2
   66       CN(IRAD,IRAD,IHGT+1)=CN(IRAD,IRAD,IHGT+1)+OLDWT*RATIO1
   67       GO TO 400
   68    21 IF(IHGT-1) 23,23,25
   69    23 IF(RADIUS(IHGT,IRAD)-RMAX*.999) 319,319,329
   70   329 CE( IRAD+1,1) =CE( IRAD+1,1)+OLDWT*RATIB2
   71       CS(IRAD,IRAD,1)=CS(IRAD,IRAD,1)+OLD*T*RATIB1
   72       GO TO 400
   73   318 CN(IRAD,IRAD,IHGT+1)=CN(IRAD,IRAD,IHGT+1)+OLDWT
   74       GO TO 400
   75   319 CS(IRAD,IRAD,1)=CS(IRAD,IRAD,1)+OLD*T
   76       GO TO 400
   77    25 CE(IRAD+1,IHGT)=CE(IRAD+1,IHGT)+OLD*T
   78       GO TO 400
      C
   79   307 IF(EVENT.NE.7) GO TO 309
   80       IF(MEDOLD.EQ.NMED) GO TO 400
   81       IF(IHGT-IHGTN) 34,36,327
   82   327 CS(IRAD,IRADN,IHGT)=CS(IRAD,IRADN,IHGT)+OLD*T
   83       GO TO 400
   84    34 CN(IRAD,IRADN,IHGTN)=CN(IRAD,IRADN,IHGTN)+OLD*T
   85       GO TO 400
   86    36 IF(IRAD-IRADN) 317,400,337
   87   317 CE(IRADN,IHGTN)=CE(IRADN,IHGTN)+OLD*T
   88       GO TO 400
   89   337 CW(IRAD,IHGT)=CW(IRAD,IHGT)+OLDWT
   90       GO TO 400
   91   309 IF(EVENT.NE.9) GO TO 303
   92       SD(IRADN,IHGTN)=SD(IRADN,IHGTN)+WATE
   93       GO TO 400
   94   303 IF(EVENT.NE.3) GO TO 305
   95       FS(IRADN,IHGTN)=FS(IRADN,IHGTN)+WATEF
   96       GO TO 400
   97   305 IF(EVENT.NE.5) GO TO 400
   98       CALL NSIGTA(IG,NMED,TSIG,PNAB)
   99       AB(IRADN,IHGTN)=AB(IRADN,IHGTN)+OLD*T*(1.0-PNAB)
  100   400 IRAD=IRADN
  101       IHGT=IHGTN
  102       RETURN
      C
  103   700 FKT=FTOTL/SWATE
  104       WRITE(6,620)NBATCH
  105   620 FORMAT(//10X,21H** BATCH NUMBER **  (,I4,2H) //)
  106       CALL REBAL (FS,CE,CW,CN,CS,AB,SD,SWATE,1,E=2,1,E=2,1,E=2,1,E=2,1,E
           1=2,1,E=2,IRX,IRY,IRXP,IRYP,NBATCH,1.0,0.0,0.0,FKT,IEND,FTOTL)
  107       RETURN
  108       END
```

Appindix IV-3

```
      SUBROUTINE REBAL(FS,FR,FL,AB,SD,@@S,VAR,EPS1,EPS2,EPS3,EPS4,
     1                 IR,IRP,NBATCH,NBDL,NBDR,EV,IEND,RAMBD)
CCC   1                 IR,IRP,NBATCH,NBDL,NBDR,EV,IEND)
      DIMENSION FS(IR),FR(IRP),FL(IRP),AB(IR),SD(IR)
      DIMENSION F(10),T(10),G(10),H(10),@@(10),ABSD(10)

        COMMON /FACTOR/ REACT(10)
C
        IEND=0
        DO 10 L=1,IR
     10 @@(L)=FS(L)/EV
        RAMOLD=@@S*EV
C
        K=1
        EVOLD1=EV
        TCR=FL(1)+FR(IRP)
        DO 15 L=1,IR
        ABSD(L)=AB(L)+SD(L)
     15 TCR=TCR+ABSD(L)
C***
        WRITE(6,600) NBATCH,EV,@@S,TCR
  600 FORMAT(///1X,'***** REBAL ROUTINE ****'/2X,'NBATCH=',I2,2X,'K(EFF)
     1=',E12.5,2X,'@@S=',E12.5,2X,'TCR+AB+SD=',E12.5/1X,'K=0')
        WRITE(6,605) (FS(L),L=1,IR)
        WRITE(6,610) (@@(L),L=1,IR)
        WRITE(6,615) (AB(L),L=1,IR)
        WRITE(6,620) (SD(L),L=1,IR)
        WRITE(6,625) (FR(L),L=1,IRP)
        WRITE(6,630) (FL(L),L=1,IRP)
  605 FORMAT(1X,2HFS,10(2X,1PE10.3))
  610 FORMAT(1X,2H@@,10(2X,1PE10.3))
  615 FORMAT(1X,2HAB,10(2X,1PE10.3))
  620 FORMAT(1X,2HSD,10(2X,1PE10.3))
  625 FORMAT(1X,2HFR,11(2X,1PE10.3))
  630 FORMAT(1X,2HFL,11(2X,1PE10.3))
C***
C
C  FORWARD ELIMINATION
C
 1000 DO 40 J=1,IR
        T(J)=ABSD(J)
        IF((J.EQ.1).AND.(NBDL.EQ.1)) GO TO 20
        T(J)=T(J)+FL(J)
     20 IF((J.EQ.IR).AND.(NBDR.EQ.1)) GO TO 25
        T(J)=T(J)+FR(J+1)
     25 IF(J.NE.1) GO TO 30
        G(J)=@@(J)/T(J)
        GO TO 35
     30 T(J)=T(J)-FR(J)*H(J-1)
        IF(ABS(T(J)).LT.1.0E-70) GO TO 55
        G(J)=(@@(J)+FR(J)*G(J-1))/T(J)
     35 H(J)=FL(J+1)/T(J)
     40 CONTINUE
C
C  BACKWARD SUBSTITUTION
C
        F(IR)=G(IR)
        IF(F(IR).LE.0.0) GO TO 55
        IRM1=IR-1
        DO 45 J=1,IRM1
        I=IR-J
        F(I)=G(I)+H(I)*F(I+1)
        IF(F(I).LE.0.0) GO TO 55
     45 CONTINUE
C***
        WRITE(6,635) K,(F(L),L=1,IR)
  635 FORMAT(1X,'K=',I3/1X,2H F,10(2X,1PE10.3))
C***
```

```
      M=IRY-LY+1
   30 WRITE(6,650) M,(FS(LX,M),LX=1,IRX)
C
  151 WRITE(6,610)
      WRITE(6,645) (I,I=1,IRX)
      DO 31 LY=1,IRY
      M=IRY-LY+1
   31 WRITE(6,650) M,(QQ(LX,M),LX=1,IRX)
      WRITE(6,615)
      WRITE(6,645) (I,I=1,IRX)
      DO 35 LY=1,IRY
      M=IRY-LY+1
   35 WRITE(6,650) M,(AB(LX,M),LX=1,IRX)
      WRITE(6,620)
      WRITE(6,645) (I,I=1,IRX)
      DO 36 LY=1,IRY
      M=IRY-LY+1
   36 WRITE(6,650) M,(SD(LX,M),LX=1,IRX)
      WRITE(6,625)
      WRITE(6,645) (I,I=1,IRX)
      DO 40 LY=1,IRY
      M=IRY-LY+1
   40 WRITE(6,650) M,(FRX(LX,M),LX=1,IRXP)
      WRITE(6,630)
      WRITE(6,645) (I,I=1,IRX)
      DO 41 LY=1,IRY
      M=IRY-LY+1
   41 WRITE(6,650) M,(FLX(LX,M),LX=1,IRXP)
      WRITE(6,635)
      DO 42 J=1,IRX
      WRITE(6,645) (I,I=1,IRX)
      DO 42 LY=1,IRYP
      M=IRYP-LY+1
   42 WRITE(6,650) M,(FRY(LX,J,M),LX=1,IRX)
      WRITE(6,640)
      DO 43 J=1,IRX
      WRITE(6,645) (I,I=1,IRX)
      DO 43 LY=1,IRYP
      M=IRYP-LY+1
   43 WRITE(6,650) M,(FLY(LX,J,M),LX=1,IRX)
      IF(N) 1000,1000,1001
C***
CALCULATION START
C
 1000 N=1
 2000 DO 100 LY=1,IRY
C
C  ADJUSTING RIGHT HAND VECTOR
      DO 45 LX=1,IRX
      QT(LX,LY)=QQ(LX,LY)
      IF(LY.EQ.1) GO TO 50
      FRYSUM=0.0
      DO 46 J=1,IRX
   46 FRYSUM=FRYSUM+FRY(J,LX,LY)*F(J,LY-1)
      QT(LX,LY)=QT(LX,LY)+FRYSUM
   50 IF(LY.EQ.IRY) GO TO 45
      FLYSUM=0.0
      DO 47 J=1,IRX
   47 FLYSUM=FLYSUM+FLY(J,LX,LY+1)*FOLD(J,LY+1)
      QT(LX,LY)=QT(LX,LY)+FLYSUM
   45 CONTINUE
C
```

```
      C  FORWARD ELIMINATION
  108    56 DO 90 J=1,IRX
  109       JMAX=J
  110       T(J)=ABSD(J,LY)
  111       IF((J.EQ.1).AND.(NBDXL.EQ.1)) GO TO 60
  112       T(J)=T(J)+FLX(J,LY)
  113    60 IF((J.EQ.IRX).AND.(NBDXR.EQ.1)) GO TO 65
  114       T(J)=T(J)+FRX(J+1,LY)
  115    65 IF((LY.EQ.1).AND.(NBDYL.EQ.1)) GO TO 70
  116       DO 58 KK=1,IRX
  117    58 T(J)=T(J)+FLY(J,KK,LY)
  118    70 IF((LY.EQ.IRY).AND.(NBDYR.EQ.1)) GO TO 75
  119       DO 59 KK=1,IRX
  120    59 T(J)=T(J)+FRY(J,KK,LY+1)
  121    75 IF(J.NE.1) GO TO 80
  122       G(J)=QT(J,LY)/T(J)
  123       GO TO 85
  124    80 T(J)=T(J)-FRX(J,LY)*H(J-1)
  125       IF(ABS(T(J)).LT.1.0E-70) GO TO 110
  126       G(J)=(QT(J,LY)+FRX(J,LY)*G(J-1))/T(J)
  127    85 H(J)=FLX(J+1,LY)/T(J)
  128       IF(RMAX-RADIUS(LY,J)*1.001) 57,57,90
  129    90 CONTINUE
      C
      C  BACKWARD SUBSTITUTION
  130    57 F(JMAX,LY)=G(JMAX)
  131       IF(G(JMAX).LE.0.0) GO TO 110
  132       IRM1=JMAX-1
  133       IF(IRM1) 100,100,96
  134    96 DO 95 J=1,IRM1
  135       I=JMAX-J
  136       F(I,LY)=G(I)+H(I)*F(I+1,LY)
  137       IF(F(I,LY).LE.0.0) GO TO 110
  138    95 CONTINUE
  139   100 CONTINUE
      C***
  140       IF(N.NE.1) GO TO 310
  141   300 DO 305 LY=1,IRY
  142       DO 305 LX=1,IRX
  143   305 FOLD(LX,LY)=F(LX,LY)
  144       N=N+1
  145       GO TO 2000
  146   310 IF(N.GT.10 ) GO TO 320
  147       E0=0.0
  148       DO 315 LY=1,IRY
  149       DO 315 LX=1,IRX
  150       TEMP=F(LX,LY)/FOLD(LX,LY)
  151       TEMP=ABS(1.0-TEMP)
  152   315 E0=AMAX1(E0,TEMP)
  153       IF(E0.GE.EPS0) GO TO 300
  154       GO TO 101
  155   320 WRITE(6,690)
  156       RETURN
      C
      C
  157   101 E1=0.0
  158       DO 105 LY=1,IRY
  159       DO 105 LX=1,IRX
  160       TEMP=1.0-F(LX,LY)
  161   105 E1=AMAX1(E1,ABS(TEMP))
  162       IF((VAR-E1).LE.EPS4) GO TO 120
      C
```

```
      C  WHOLE SYSTEM BALANCING
  163   110 TEMP=QQS/TCR
  164       DO 115 LY=1,IRY
  165       DO 115 LX=1,IRX
  166   115 RFACT(LX,LY)=TEMP
  167       EV=TEMP*EVOLD1
  168       RAMBD=EV*QQS
  169       WRITE(6,660) K,TEMP
  170       GO TO 165
      C
  171   120 IF((K.NE.1).OR.(NBATCH.EQ.1)) GO TO 125
  172       WRITE(6,665) EV,EVOLD
  173       E2=(EVOLD-EV)/EVOLD
  174       E2=ABS(E2)
  175       IF((E1.GE.EPS1).OR.(E2.GE.EPS2)) GO TO 125
  176       IEND=100
  177       RETURN
      C
  178   125 RAMBD=0.0
  179       DO 130 LY=1,IRY
  180       DO 130 LX=1,IRX
  181       RFACT(LX,LY)=F(LX,LY)*RFACT(LX,LY)
  182       FS(LX,LY)=F(LX,LY)*FS(LX,LY)
  183   130 RAMBD=RAMBD+FS(LX,LY)
  184       EV=RAMBD/QQS
  185       WRITE(6,670) EV,RAMBD,RAMOLD
  186       DO 135 LY=1,IRY
  187       DO 135 LX=1,IRX
  188   135 QQ(LX,LY)=FS(LX,LY)/EV
  189       IF(NBATCH.LE.30) GO TO 140
  190       GO TO 165
      C
  191   140 TEMP=(RAMOLD-RAMBD)/RAMOLD
  192       TEMP=ABS(TEMP)
  193       IF(TEMP.LT.EPS3) GO TO 165
  194       RAMOLD=RAMBD
  195       DO 145 LY=1,IRY
  196       DO 145 LX=1,IRX
  197       FRX(LX+1,LY)=F(LX,LY)*FRX(LX+1,LY)
  198       FLX(LX,LY)=  F(LX,LY)*FLX(LX,LY)
  199       DO 146 J=1,IRX
  200       FRY(LX,J,LY+1)=F(LX,LY)*FRY(LX,J,LY+1)
  201   146 FLY(LX,J,LY)=F(LX,LY)*FLY(LX,J,LY)
  202       AB(LX,LY)=   F(LX,LY)*AB(LX,LY)
  203       SD(LX,LY)=   F(LX,LY)*SD(LX,LY)
  204   145 ABSD(LX,LY)=F(LX,LY)*ABSD(LX,LY)
      C***
  205       WRITE(6,605)
  206       WRITE(6,645) (I,I=1,IRX)
  207       DO 150 LY=1,IRY
  208       M=IRY-LY+1
  209   150 WRITE(6,650) M,(FS(LX,M),LX=1,IRX)
  210       GO TO 151
      C***
      C
  211  1001 K=K+1
  212       IF(K.LE.10 ) GO TO 1000
  213       WRITE(6,675)
  214       GO TO 110
  215   165 DO 170 LY=1,IRY
  216       DO 170 LX=1,IRX
  217   170 RFACT(LX,LY)=RFACT(LX,LY)*EVOLD1/EV

  218       EVOLD=EV
  219       WRITE(6,680)
  220       WRITE(6,645) (I,I=1,IRX)
  221       DO 175 LY=1,IRY
  222       M=IRY-LY+1
  223   175 WRITE(6,650) M,(RFACT(LX,M),LX=1,IRX)
  224       RETURN
  225       END
```

Appendix IV-4

```
      SUBROUTINE REBAL3(FS,FRX,FLX,FRY,FLY,AB,SD,QQS,VAR,EPS0,EPS1,EPS2,
     1                  EPS3,EPS4,IRX,IRY,IRXP,IRYP,NBATCH,NBDXL,NBDXR,
     2                  NBDYL,NBDYR,EV,IEND,RAMBD)
      DOUBLE PRECISION  RADIUS,HIGHT,HMAX,RMAX,BIG
      DIMENSION FS( 10,10 ),FRX(  11,10 ),FLX(  11,10 ),FRY(10,10,11),
     1          FLY(10,10,11),AB( 10,10 ),SD(10,10)
      DIMENSION F(10,10),FOLD(10,10),QQ(10,10),ABSD(10,10),
     1          T(10),G(10),H(10),QT(10,10)
      COMMON /FACTOR/ REACT(10,10)
      COMMON/DOMIN1/RADIUS(20,20),HIGHT(20),HMAX,RMAX,BIG,NORAD,NOHGT
C
C
  600 FORMAT(///1X,'***** REBAL ROUTINE *****'/2X,'NBATCH=',I2,2X,'K(EFF
     1)=',E12.5,2X,'QQS=',E12.5,2X,'TCR+AB+SD=',E12.5/1X,'K=0')
  605 FORMAT(/1X,3(8(1H-),' FS  ',8(1H-)))
  610 FORMAT(/1X,3(8(1H-),' QQ  ',8(1H-)))
  615 FORMAT(/1X,3(8(1H-),' AB  ',8(1H-)))
  620 FORMAT(/1X,3(8(1H-),' SD  ',8(1H-)))
  625 FORMAT(/1X,3(8(1H-),' FRX ',8(1H-)))
  630 FORMAT(/1X,3(8(1H-),' FLX ',8(1H-)))
  635 FORMAT(/1X,3(8(1H-),' FRY ',8(1H-)))
  640 FORMAT(/1X,3(8(1H-),' FLY ',8(1H-)))
  645 FORMAT(2X,'LY/LX',4X,11I9/12)
  650 FORMAT(3X,I1,10(2X,1PE10.3))
  660 FORMAT(1X,'K=',I3/1X,'#HOLE SYSTEM BALANCING.   F=',E12.5)
  665 FORMAT(1X,'K(EFF)... NEW=',E12.5,2X,'OLD=',E12.5)
  670 FORMAT(1X,'E.V.=',E12.5,2X,'RAMBDA... NEW=',E12.5,2X,'OLD=',E12.5)
  675 FORMAT(1X,'*** NOT CONVERGED (K-ITERATION)')
  680 FORMAT(/1X,10(1H+),' REACT ',10(1H+))
  690 FORMAT(1X,'*** NOT CONVERGED (N-ITERATION)')
C
      IEND=0
      N=0
      DO 10 LY=1,IRY
      DO 10 LX=1,IRX
      F(LX,LY)=1.0
      FOLD(LX,LY)=1.0
   10 QQ(LX,LY)=FS(LX,LY)/EV
      RAMOLD=QQS*EV
C
      K=1
      EVOLD1=EV
      TEMP1=0.0
      DO 15 LY=1,IRY
      DO 16 LX=1,IRX
      IF(RMAX-RADIUS(LY,LX)*1.001) 17,17,16
   16 CONTINUE
   17 TEMP1=TEMP1+FRX(LX+1,LY)
   15 TEMP1=TEMP1+FLX(1,LY)
      TEMP2=0.0
      DO 20 LX=1,IRX
   20 TEMP2=TEMP2+FLY(LX,LX,1)+FRY(LX,LX,IRYP)
      TCR=TEMP1+TEMP2
      DO 25 LY=1,IRY
      DO 25 LX=1,IRX
      ABSD(LX,LY)=AB(LX,LY)+SD(LX,LY)
   25 TCR=TCR+ABSD(LX,LY)
C***
      WRITE(6,600) NBATCH,EV,QQS,TCR
      WRITE(6,605)
      WRITE(6,645) (I,I=1,IRX)
      DO 30 LY=1,IRY
```

```
      C
   53       E1=0.0
   54       DO 50 L=1,IR
   55       TEMP=1.0-F(L)
   56    50 E1=AMAX1(E1,ABS(TEMP))
   57       IF((VAR-E1).LE.EPS4) GO TO 65
      C
   58    55 TEMP=@@S/TCR
   59       DO 60 L=1,IR
   60    60 RFACT(L)=TEMP
   61       EV=TEMP*EVOLD1
      C***
   62       WRITE(6,640) K,TEMP
   63   640 FORMAT(1X,'K=',I3/1X,'*HOLE SYSTEM REBALANCING.   F=',E12.5)
      C***
   64       GO TO 95
      C
   65    65 IF((K.NE.1).OR.(NBATCH.EQ.1)) GO TO 70
      C***
   66       WRITE(6,645) EV,EVOLD
   67   645 FORMAT(1X,'K(EFF)... NEW=',E12.5,2X,'OLD=',E12.5)
      C***
   68       E2=(EVOLD-EV)/EVOLD
   69       E2=ABS(E2)
   70       IF((E1.GE.EPS1).OR.(E2.GE.EPS2)) GO TO 70
   71       IEND=100
   72       RETURN
      C
   73    70 RAMBD=0.0
   74       DO 75 L=1,IR
   75       RFACT(L)=F(L)*RFACT(L)
   76       FS(L)=F(L)*FS(L)
   77    75 RAMBD=RAMBD+FS(L)
      C
   78       EV=RAMBD/@@S
      C***
   79       WRITE(6,650) EV,RAMBD,RAMOLD
   80   650 FORMAT(1X,'E.V.=',E12.5,2X,'RAMBDA... NEW=',E12.5,2X,'OLD=',E12.5)
      C***
   81       DO 80 L=1,IR
   82    80 @@(L)=FS(L)/EV
      C
   83       IF(NBATCH.LE.30) GO TO 85
   84       GO TO 95
      C
   85    85 TEMP=(RAMOLD-RAMBD)/RAMOLD
   86       TEMP=ABS(TEMP)
   87       IF(TEMP.LT.EPS3) GO TO 95
   88       RAMOLD=RAMBD
      C
   89       DO 90 L=1,IR
   90       FR(L+1)=F(L)*FR(L+1)
   91       FL(L)=F(L)*FL(L)
   92       AB(L)=F(L)*AB(L)
   93       SD(L)=F(L)*SD(L)
   94    90 ABSD(L)=F(L)*ABSD(L)
      C***
   95       WRITE(6,605) (FS(L),L=1,IR)
   96       WRITE(6,610) (@@(L),L=1,IR)
   97       WRITE(6,615) (AB(L),L=1,IR)
   98       WRITE(6,620) (SD(L),L=1,IR)
   99       WRITE(6,625) (FR(L),L=1,IRP)

  100       WRITE(6,630) (FL(L),L=1,IRP)
      C***
      C
  101       K=K+1
  102       IF(K.LE.100) GO TO 1000
      C***
  103       WRITE(6,655)
  104   655 FORMAT(1X,'*** NOT CONVERGED (K-ITERATION)')
      C***
  105       GO TO 55
  106    95 DO 100 L=1,IR
  107   100 RFACT(L)=RFACT(L)*EVOLD1/EV
  108       EVOLD=EV
      C***
  109       WRITE(6,660) (RFACT(L),L=1,IR)
  110   660 FORMAT(1X,5HRFACT,10(2X,1PE10.3))
      C***
  111       RETURN
  112       END
```