MITNE-216
c. 2

AN IMPROVED LONG RANGE FUEL MANAGEMENT PROGRAM

by

C. L. Beard

May, 1978

Massachusetts Institute of Technology
Department of Nuclear Engineering
Cambridge, Massachusetts

Support provided by:
Westinghouse Electric Corporation
Yankee Atomic Electric Company

ABSTRACT

A new procedure has been developed for rapid and accurate long range fuel management studies. A computer program, FLAC, was written employing this new procedure. The program has been verified and qualified for use on large, commercial, nuclear reactors. Sensitivity studies have been performed to determine the best methods for use of the program.

The procedure is derived from the FLARE model of nodal coupling. The nodal equations are combined and averaged over regions to generate coupled equations on a region-wise basis. This set of coupled equations and its corresponding eigenvalue is solved to determine the relative region fission sharings and the core average k_{eff}.

The FLAC program automatically performs the cycle depletions and has the capability of determining (1) cycle burnup, (2) end of cycle k_{eff} or (3) the required feed enrichment. Up to 20 sequential cycles or equilibrium cycle calculations can be performed. A significant number of options have been included in the program for easy utilization without requiring detailed knowledge of future cores. For example, the program can generate data pertinent to either a ring interior or a checkerboard interior assembly arrangement. The ability to use basic loading pattern data is a significant improvement over the currently used weighted k procedures. And the ability to automatically generate the loading pattern data provides for an ease of usage unattainable with multi-dimensional calculations. Thus the FLAC program allows one to do detailed fuel management studies with a simplicity of input preparation.

ACKNOWLEDGEMENTS

This report has been submitted in partial fulfillment of the requirements for the Master of Science degree at MIT. Professor David D. Lanning and Professor Michael J. Driscoll served as thesis advisors and their time, support and assistance is appreciated.

I gratefully acknowledge the support given me by the Westinghouse Electric Corporation. Without its support I would have never had this opportunity. I would also like to acknowledge the financial and technical support given by the Yankee Atomic Electric Company for the completion of this research. Special thanks are due to Ed Pilat, whose ideas spurred this project. Of course, this project could never have been completed without the support and help of the MIT Staff and students. Special thanks are also in line for Rachel Morton whose invaluable aid greatly assisted in my familiarization with the MIT Computer system. Last, but not least, is my gratitude for my wife, Donna, and my children, Sheryl and Patrick, for their willingness to temporarily relocate our home and especially for always providing me with the needed diversion to get away from it all.

TABLE OF CONTENTS

LIST OF TABLES

FIGURES

CHAPTER I

INTRODUCTION

Nuclear fuel management is one of the major responsibilities of a utility with nuclear power plants. The objective is to obtain the maximum utilization of the fuel while meeting scheduling and safety constraints. There are currently a number of computational tools that are used in this area, ranging from the trivial (5,7,8,12) to the complicated (3,11). However, the trivial require significant normalization and the complicated are very costly and time-consuming. Therefore, a computer program has been developed embodying the benefits of both the trivial and the complicated tools.

The model as developed makes use of the FLARE (4) equations on a regionwise basis. Thus it keeps the regionwise data of the trivial codes but solves for the k_{eff} and power distribution like the complicated codes.

A computer code, FLAC, has been written by using the model as described in this report. The program incorporates many options and automated input preparation to enable it to be easily used. It has been thoroughly tested, verified and qualified. Its usefulness has been demonstrated with numerous studies which can be done quickly and easily.

The main thrust of this development has been the modeling of pressurized water reactor cores and all the test cases on the computer program have been with PWR's. However, it is felt that these methods could be extended to boiling water reactors, but it is left for future work.

I.1 Background

Currently there are many ways for doing long range fuel management varying from the trivial to the very difficult and complex. The FLAC program has been developed to lie between the simple methods and the difficult and time consuming methods.

The simple methods (5,7,8,12) are based on variations of the general formula:

$$k_{av} = f^{-1}\left\{\left[\sum_{i=1}^{N} w_i f(k_i)\right] / \sum_{i=1}^{N} w_i\right\} \tag{1.1}$$

where

- k_{av} is the core average k,
- N is the number of regions
- k_i is the region average k and is a function of burnup and enrichment
- w_i is a weight for region i
- $f(.)$ is a general function which has an inverse $f^{-1}(.)$

The simple linear reactivity model assumes that $f(k) = k$ and that $w_i = n_i$, the number of assemblies in region i. Other models use a linear reactivity model but use a weighting function equal to the region power times the number of assemblies.

The big drawback to all of these simple methods is the need to know the region-wise power sharings to perform the calculation. Even if the weight functions do not include the power, it is still required to determine the end of cycle region-wise burnups. Thus significant engineering judgement is required or an equivalent multi-dimensional calculation.

This greatly increases the difficulty of use, especially for novel problems. Procedures of fitting the power sharings to various parameters have been tried, but these at best yield only a first approximation to the actual values. There is too much coupling in the core to adequately determine the power sharings with a simple fit. Also, the simple calculations can not properly factor in radial core leakage.

There are a variety of complicated methods for long range fuel management (3,11), but they all require dimensional depletion calculations. To do these depletions, loading patterns are required for each cycle. Therefore, in addition to the large computer expenditure, a considerable amount of man-time is required to setup these models and to find suitable loading patterns. These models usually give accurate results, but unfortunately reality seldom follows the desired path. Due to some circumstance such as plant maintenance and/or operating requirements, the actual fuel management scheme can not be followed as planned. Then all the fuel management work is in error because a base point has now been changed. Therefore, although it is possible to accurately predict core behavior for a number of cycles, it is very costly and quite likely to be invalidated by uncontrollable events. Therefore, there is little incentive to perform these detailed calculations.

I.2 Summary

The FLAC computer program has been written to provide a long range fuel management tool which generates results of the desired accuracy with a minimum of effort and cost. This report describes the model, the verification, the method of use and provides information about the computer program.

Chapter 2 describes the basic model used in FLAC as well as the many options available in the program. These options include automated generation of the fuel shuffling data from cycle to cycle and the edge correspondence between regions for either a ring or a checkerboard loading pattern. Also discussed is the interpolation scheme, the matrix solution procedure and the various iterative search procedures employed by the program.

The third chapter discusses the verification and qualification of the computer code and the sensitivity to the important input parameters. The verification demonstrates that the code has been properly programmed as specified in the model. The qualification shows that the program is applicable for long range fuel management studies for large PWR cores. The results of sensitivity analysis can be used to determine the importance of the various input parameters as an aid in deciding how much effort should be put into input generation.

Chapter 4 presents the recommended method of use of the FLAC computer program based on the previous qualification and sensitivity analyses. The method is described and the anticipated accuracy is defined. The conclusions of the study are also given along with many ideas for future work.

There are two appendices to this report, the first is a user's manual for the FLAC program and the second is a listing of the program. The user's manual includes a detailed input description and a discussion of the output edits. Sample input and output listings are given as an aid to understanding. Also included is a section giving programmer's information which includes descriptions of the various variables and the subroutines in the program.

As an aid to understanding this report a table of nomenclature has been included as Table 1-1.

TABLE 1-1

NOMENCLATURE

Symbol	Definition
α	Average albedo for an assembly
BOC	Beginning of cycle
BP	Burnable poison
Bu	Burnup
BWR	Boiling water reactor
EFPH	Effective full power hours
EOC	End of cycle
ε	Initial enrichment
$F_{\Delta H}$	Hot channel peaking factor
GWD/MTM	Giga-wall days per metric ton metal
K	Energy release per fission
k_{eff}	Core effective k
k_{∞}	Infinite core k
λ	Fundamental eigenvalue
MWD/MTM	Megawatt days per metric ton metal
n	Number of edges between regions
N	Number of regions or number of assemblies
ν	Neutrons produced per fission
ω	Over-relaxation parameter
PWR	Pressurized water reactor
S	Fission Source
Σ_f	Macroscopic fission cross section
W	The probability that a neutron produced in one assembly will be absorbed
w/o	Weight per cent, generally w/o of U-235 in the uranium fuel

CHAPTER 2

ANALYTICAL METHODS

FLAC is a computer program which is based on very simple, fundamental assumptions. As such, it is very easy to comprehend the analytical model underlying the program. However, in order to make FLAC a very useful program, many different conveniences were built into the code. These include various interpolation schemes, search procedures and input generation. The basic models and assumptions underlying all of these aspects are presented in this section.

2.1 Neutronics Model

Starting with the basic FLARE (4) equation:

$$S_j = \frac{\frac{k_{\infty j}}{\lambda} \sum_m S_m W_{mj}}{1 - \frac{k_{\infty j}}{\lambda} W_{jj}} \tag{2.1}$$

where S_j = rate of production of fission energy neutrons at node; where each node is one fuel assembly

W_m = probability that a neutron born at node m will ultimately be absorbed at node j

$k_{\infty j}$ = k_∞ at node j including xenon and axial affects

λ = fundamental eigenvalue

Also, we know that:

$$W_{jj} = 1 - \sum_m W_{jm} - \sum_{m'}' (1 - \alpha_{m'}) W_{jm'} \tag{2.2}$$

where:

m' are the edges on the exterior

m are the edges on the interior

and $\alpha_{m'}$ is the albedo on an exterior edge.

If W_{mj} is independent of j, then $W_{mj} = W_m$. Also, $m + m' = 4$ for every node. Therefore, equation 2.2 can be rewritten as:

$$W_{jj} = 1 - 4W_j + \sum_{m'} \alpha_{m'} W_j \tag{2.3}$$

Combining equations 2.1 and 2.3 gives:

$$S_j = \frac{\frac{k_{\infty j}}{\lambda} \sum_m S_m W_m}{1 - \frac{k_{\infty j}}{\lambda} [(1 - 4W_j) + \sum_{m'} \alpha_{m'} W_j]} \tag{2.4}$$

which can be rewritten as:

$$[\frac{\lambda}{k_{\infty j}} - (1 - 4W_j) + \sum_{m'} \alpha_{m'} W_j] S_j = \sum_m S_m W_m \tag{2.5}$$

Assume that $k_{\infty j}$ and W_j are the same for each node in region I. This assumption implies that all the assemblies in a region are identical. That is they have the same enrichment, burnup and burnable poison content. Then summing over all the nodes in region I gives:

$$N_I S_I [\frac{\lambda}{k_{\infty I}} - (1 - 4W_I)] + \alpha_I S_I W_I n_{I\alpha} = \sum_{J=1}^{N} S_J W_J n_{IJ} \tag{2.6}$$

for $I = 1, 2, \ldots, N$

where:

N_I is the number of assemblies in region I

S_I is the average fission source in region I

W_I is the average absorption probability in region I

α_I is the average albedo in region I

n_{IJ} is the number of edges of region I adjacent to region J

$n_{I\alpha}$ is the number of assembly edges in region I on the exterior and

N is the number of regions.

Note that $n_{IJ} = n_{JI}$ and that $4N_I = \sum_{J=1}^{N} n_{IJ} + n_{I\alpha}$. Equation 2.6 can be rewritten as:

$$\{[(1 - 4W_I) - 4\alpha_I f_{I\alpha} W_I]k_{\infty I} - \lambda\}S_I + \sum_{J=1}^{N} (4W_J f_{IJ} k_{\infty I})S_J = 0$$

$$\text{for } I = 1, 2, \ldots, N$$

where

$$f_{I\alpha} = \frac{n_{I\alpha}}{4N_I} = \text{fraction of edges on the exterior}$$

$$f_{IJ} = \frac{n_{IJ}}{4N_I} = \text{fraction of edges in region I adjacent to region J}$$

$$f_{I\alpha} + \sum_{J=1}^{N} f_{IJ} = 1.0 \qquad (2.8)$$

If one defines

$$\beta_I = [(1 - 4W_I) - 4\alpha_I f_{I\alpha} W_I]k_{\infty I} \qquad (2.9)$$

and $\gamma_{IJ} = 4W_J f_{IJ} k_{\infty I}$ (2.10)

Then Equation 2.7 can be simply written as:

$$(\beta_I - \lambda)S_I + \sum_{J=1}^{N} \gamma_{IJ}S_J = 0 \qquad \text{for } I = 1, 2, \ldots, N \tag{2.11}$$

which is a standard eigenvalue problem.

The matrix formulation for equation 2.11 is:

$$\begin{bmatrix} (\beta_1 + \gamma_{11})-\lambda & \gamma_{12} & \cdots & \gamma_{1N} \\ \gamma_{21} & (\beta_2 + \gamma_{22})-\lambda & \cdots & \gamma_{2N} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \gamma_{N1} & \gamma_{N2} & \cdots & (\beta_N + \gamma_{NN})-\lambda \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ \cdot \\ \cdot \\ \cdot \\ S_N \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \tag{2.12}$$

Solving for the largest eigenvalue will give the approximate core eigenvalue. Its eigenvector will give the approximate assembly average source terms.

The power sharing can then be found from

$$P_I = \left(\frac{\kappa\Sigma_f}{\nu\Sigma_f}\right)_I S_I = \left(\frac{\kappa}{\nu}\right)_I S_I \tag{2.13}$$

where $\left(\frac{\kappa}{\nu}\right)_I$ is the number of neutrons per fission divided by the average energy production per fission for region I.

The power sharings are then normalized so that the volume weighted average is unity.

2.2 Matrix Solution

The matrix generated is a very small (on the order of 3 x 3 to 7 x 7) matrix of the same form as in the FLARE program. We know that it is a positive definite matrix within the context of its representation of a reactor core and its representative parameters. The largest eigenvalue will be distinct and positive and its eigenvector positive. Therefore, a straight power iteration with an initial unity estimate for the eigenvector will generate the desired eigenvalue and eigenvector.

As an aid to convergence, an over-relaxation procedure is used of the form:

$$S^{i+1} = S^{i} + \omega([A]S^{i} - S^{i}) \tag{2.14}$$

where ω is the over-relaxation parameter. The default value built into the code for ω is 1.80.

The iteration proceeds until convergence which is measured by two parameters:

$$\Delta_1 = \max\left(|S_j^{i+1} - S_j^{i}|\right)$$

and

$$\Delta_2 = |\lambda^{i+1} - \lambda^{i}|.$$

Convergence is assumed if $\Delta_1 \leq 1 \times 10^{-3}$ and $\Delta_2 \leq 1 \times 10^{-5}$. With the small matrices encountered, a maximum of 10-15 iterations are usually required.

To insure the elimination of round-off, the matrix solution routine is written in double precision on the IBM 360.

2.3 Generation of Tabular Data

A significant amount of data must be input in tabular form to the program. Provisions have been made in the program to simplify this input by using default data and fitting functions similar to the ones in FLARE.

A default set of burnups are given in the code and are the values 0, 4, 8, 12, ..., 48 GWD/MTU. A default set of axial leakage factors are also built into the code for a 12 foot PWR. These are presented in Figure 2.1. Corrections for cores other than 12 foot are made by the expression:

$$\text{BIAS} = \left(\frac{H + 0.5}{12.5}\right)\left(\text{BIAS}(12') - 1.0\right) + 1.0 \qquad (2.15)$$

where

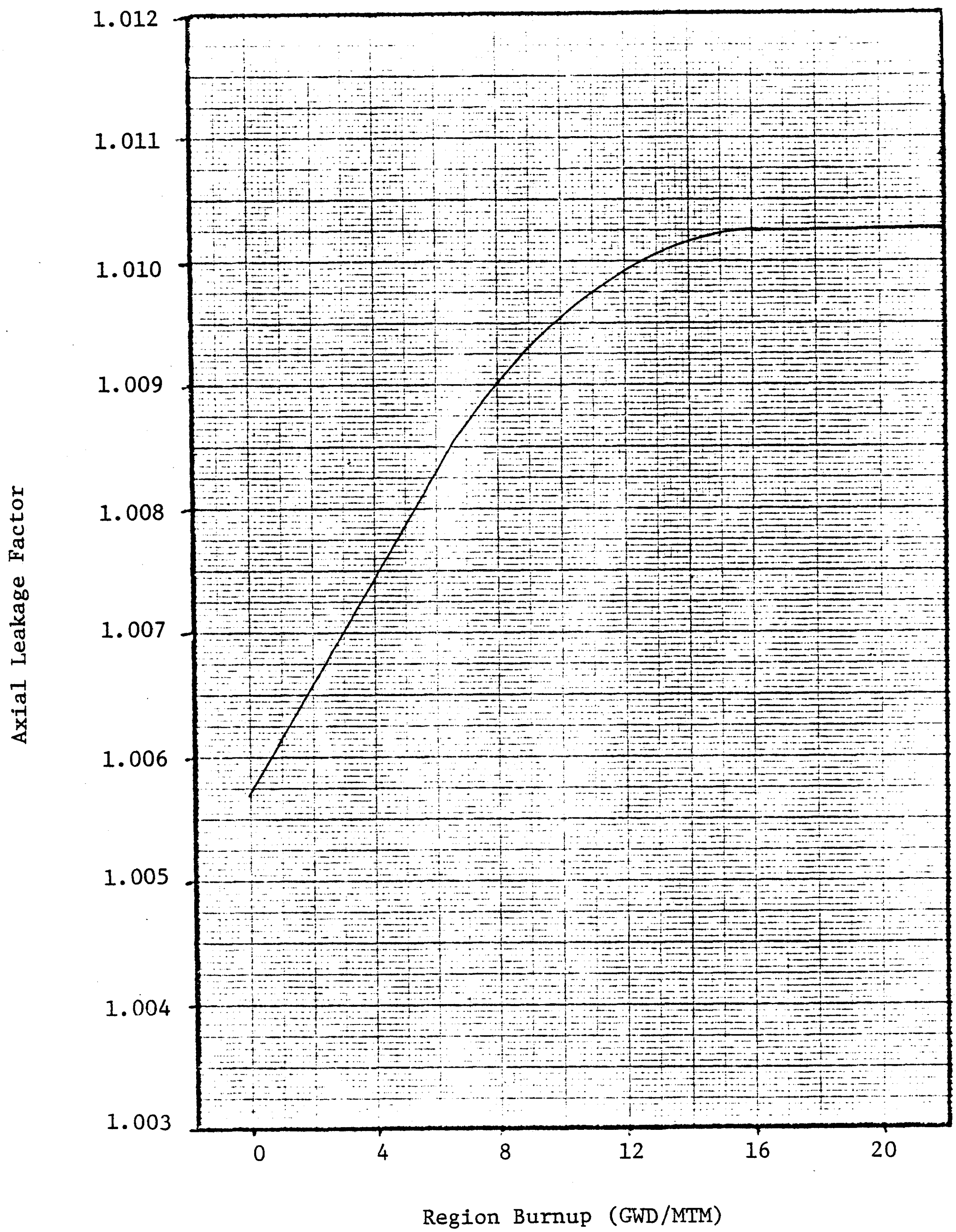
H is the core height in feet and 0.5 feet is the approximate reflector savings. (This is about 7.6 cm for each end of the core).

The k_∞ may be entered in tabular form or by coefficients of a function. When they are input in tabular form, care must be taken to ensure that the different sets for different enrichments have the same burnup correspondence. The k_∞ entered should correspond to a full power k_∞ including equilibrium xenon, but should not include any soluble poison.

When FLARE type input is used to generate this data, the functional form may be used. The function is given by:

Figure 2.1

Axial Leakage Factor



$$k_\infty = k_\infty^0 Z_1 Z_2$$

$$k_\infty^0 = B_1$$

$$Z_1 = 1 - \frac{\Delta\rho_{xe}}{\rho} - \frac{\Delta\rho_{Dop}}{\rho}$$

$$\frac{\Delta\rho_{Dop}}{\rho} = B_2$$

$$\frac{\Delta\rho_{xe}}{\rho} = B_3(1 + B_4 Bu)$$

$$Z_2 = 1 - B_5 Bu + B_6 Bu^2 + B_7 Bu^3 + B_8 Bu^4 \tag{2.16}$$

where Bu is burnup in GWD/MTU and ρ is the reactivity for the different variables (xenon, doppler).

The burnable poison worth as a function of burnup is also entered. The nominal BP worth is the worth of one rod in one assembly. This can be tabulated or entered in a functional form. The function form is given by

$$BP(Bu) = B_1(1 + B_2 Bu + B_3 Bu^2 + B_4 Bu^3 + B_5 Bu^4) \tag{2.17}$$

where Bu is the burnup in GWD/MTU. It is usually a fairly good approximation to assume that the BP worth is relatively insensitive to the enrichment.

If a dataset for a fuel cost calculation is to be prepared, then isotopic data as a function of burnup and enrichment is required. The data needed are the uranium depletion, U-235 enrichment and fissile plutonium production. This information may be input or standard default data will be used. This default data is the same as that built into the MUDEL (1,5) code and consists of three empirical relations. For a given burnup (GWD/MTU) and enrichment (w/o) the following equations are used.

$$u/U_o = \exp[-0.00159\ Bu(1 - 4.1 \times 10^{-3}\ Bu)] \tag{2.18}$$

$$U_{25} = \varepsilon \exp[-0.1162 \frac{Bu}{\varepsilon^{0.965}} (1 + 2.75 \times 10^{-5}\ Bu^2\ \varepsilon)] \tag{2.19}$$

$$10^3\ Pu/U_o = 4.795\ \varepsilon^{0.4}\ [1 - \exp(-0.1425 \frac{B_4}{\varepsilon^{0.6}})] \tag{2.20}$$

The accuracy of these expressions are discussed in Reference (1) and will not be reproduced here.

2.4 Generation of Shuffle

The program can on option automatically determine the shuffling scheme from cycle i-1 to cycle i. Two options are presently available and they deal only with the treatment of the center assembly. The shuffling is done very systematically. The feed fuel is assumed in the highest number regions. Then the regions from the previous cycle are inserted starting from the highest number region in the previous cycle. All the assemblies in each region are used except for the last region (excluding the center assembly) which uses only the number of assemblies needed. When there is a center assembly of a different region, two options are available. The first option uses an assembly from the previous cycle. The other option uses an assembly discharged from cycle 1.

Explicit shuffling schemes must be input if the regions are not to be progressed in sequential order or if assemblies are to be reinserted after sitting out of the core for one or more cycles.

2.5 Generation of Edge Data

FLAC is different from all other programs in the need for edge data. That is the number of assembly edges adjacent to assemblies of each region and the periphery. Given a loading pattern, one can carefully count and generate this information. However, one of the attributes of FLAC is the ability to perform future cycle calculations without requiring loading patterns. Therefore, coding was generated to enable the automatic computation of this edge data for two different loading pattern schemes. The two schemes are commonly referred to as the ring pattern and the checkerboard pattern. The ring pattern assumes that the regions are located in symmetric rings starting with the first region at the center. The checkerboard pattern assumes that the highest number regions form a peripheral ring with the interior regions arranged in a checkerboard distribution. The coding developed will not duplicate the edge data for a specific loading pattern, but it is general coding which will closely approximate the information for any shuffling procedure which is used.

Both schemes use the same procedure for the peripheral assemblies. The only information needed is the number of assemblies across the core and the number of assemblies on the periphery with edges on the exterior. The code assumes that the highest number region will be furthest out in the core. Therefore, it starts with those assemblies with two edges on the exterior, then goes to those assemblies with one edge on the exterior and then those assemblies on the periphery at a corner with no edges on the exterior. The number of edges on the exterior is simply:

$$n_{\alpha} = 4N_{row} \tag{2.21}$$

where N_{row} is the number of assembly rows. The number of peripheral assemblies with two edges on the exterior is given by:

$$n_{2\alpha} = n_{\alpha} - N_p \tag{2.22}$$

where N_p is the number of assemblies on the periphery. The number of assemblies at an inside corner of the periphery with no edges on the exterior is given by:

$$n_{0\alpha} = n_{\alpha} - N_p - 4 \tag{2.23}$$

and of course the number of assemblies in the peripheral region with one edge on the exterior is given by:

$$n_{1\alpha} = n_{\alpha} - n_{2\alpha} \tag{2.24}$$

The program begins by segregating the peripheral assemblies with two edges on the exterior. They are assumed to have no common edges. The program then segregates out the peripheral assemblies with one edge on the exterior. Edges are assumed to exist between the assemblies with one and two edges on the exterior. However, a maximum is set of one edge for those assemblies with one edge on the exterior and two edges for those assemblies with two edges on the exterior. The assemblies on the periphery with one edge on the exterior should by definition only have one edge left to face the interior. Therefore, any excess edges are assumed to be between those assemblies themselves.

The program next fills in the inside corner assemblies. These assemblies have two edges adjacent to peripheral assemblies and two edges on the interior. Therefore, starting with any excess edges left from the assemblies with two edges on the exterior the correspondence is assigned to have two edges of each interior corner assembly adjacent to some peripheral assembly. This completes the core periphery.

The checkerboard pattern is generated assuming a uniform distribution of the interior regions. If there is a region with a center assembly, it is arbitrarily assumed to adjoin region 3 assemblies. The peripheral assemblies are assumed to adjoin the interior regions in direct proportion to the number of assemblies in each region. Similarly, the interior regions are assumed to be adjacent to the other interior regions in direct proportion to the number of assemblies in each region. Any remaining edges are assumed to be adjacent to assemblies of its same region. The logic for the checkerboard is ideal and therefore will be an approximation to any actual loading pattern.

The ring pattern on the interior is a little more complex. One must first define the rings. A standard definition is used which gives the number of assemblies within a ring and all smaller rings. The rings are numbered sequentially from the center with each additional row of assemblies being a new ring. Also, different formulas were developed for an even number of assemblies versus an odd number of assemblies in a core.

For an odd number of assemblies, the number of assemblies with a ring is given by:

$$N_I = 2(I-1)(I+2) + 1 \qquad I = 1, 2, 3, \ldots \tag{2.25}$$

This was found to give a very accurate estimate of the number of assemblies in each ring.

For an even number of assemblies, no direct formula was found, so an area approach was used:

$$N_I = \pi I^2 \qquad I = 1, 2, 3, \ldots \tag{2.26}$$

which was then rounded off to the nearest eight assemblies (or four for the inner-most ring).

In determining the edge correspondence between rings, many variables have to be considered. It was assumed that assemblies in ring I would first fill any vacant edge positions in ring I-1. But only one edge per assembly was allowed to be adjacent to ring I-1. Then, each ring must have eight additional edges on the outside of the ring since it is larger. Any edges left over are assumed to be within assemblies in the same ring. In meshing with the periphery, it is treated as any other ring interface.

2.6 Interpolation Scheme

Many of the values used by the program are input in tabular form and require interpolation or extrapolation. Whenever possible, the interpolation scheme used is the Lagrange 3 point interpolation. The interpolation of $f(x_i)$ given a fixed value of x is given by:

$$f(x) = \alpha_1 f(x_i) + \alpha_2 f(x_{i+1}) + \alpha_3 f(x_{i+2}) \qquad (2.27)$$

where

$$\alpha_1 = \frac{(x_{i+1} - x)(x_{i+2} - x)}{(x_{i+1} - x_i)(x_{i+2} - x_i)}$$

$$\alpha_2 = \frac{(x_i - x)(x_{i+2} - x)}{(x_i - x_{i+1})(x_{i+2} - x_{i+1})}$$

and $$\alpha_3 = \frac{(x_i - x)(x_{i+1} - x)}{(x_i - x_{i+2})(x_{i+1} - x_{i+2})}$$

The values x_i, x_{i+1} and x_{i+2} are chosen so that the value x is encompassed.

For extrapolation, a simple linear model is used to assure that no change in slope occurs in the function being extrapolated. The linear extrapolation is given by:

$$f(x) = \alpha_1 f(x_i) + \alpha_2 f(x_{i+1}) \tag{2.28}$$

where

$$\alpha_1 = \frac{(x_{i+1} - x)}{(x_{i+1} - x_i)}$$

$$\alpha_2 = 1 - \alpha_1$$

The values x_i and x_{i+1} are the closest to the desired variable x.

When the independent variable is burnup, it is assumed that the table is monotonically increasing and that there are at least three burnups tabulated. The fit will be made using one burnup below the desired value and two points above it, if possible.

When the independent variable is enrichment, a more complicated scheme is used. The interpolation is made only over those of the same "fuel type" as the desired enrichment. Thus it is possible to separate different fuel types such as plutonium of different recycle modes, thorium, stainless clad, larger hydrogen to uranium ratio, etc. The program makes no implicit assumption on the number of enrichments available for interpolation or any sequential order. The program will try to determine three points in the table - the closest value greater than the desired value, the closest value less than the desired value and the next closest value to the desired value on either side. This will ensure the best three points to use in the 3 point Lagrange interpolation. Note that for extrapolation or when only two enrichments of the same type are given, only two points are found and a linear extrapolation is used. When only one enrichment is in the table, a constant is generated independent of enrichment.

To insure proper interpolation, care must be taken in generating the table. Enrichments of the same type should not be within 0.10 w/o of each other. For close enrichments, secondary variables may distort the true relationship with enrichment.

When the interpolation is to be made over both enrichment and burnup, a two-step procedure is used. The three enrichments and three burnups which are to be used in the interpolation are found as described above. Then for each reference enrichment, an interpolation is made to yield the dependent variable at the specified burnup for that reference enrichment. Then the second step consists of interpolating the values at the desired burnup and reference enrichments to give the value at the desired burnup and enrichment.

2.7 Burnup Calculation

To determine cycle lifetime a burnup calculation must be performed. Three different options are available - straight depletion with no control, straight depletion with uniform control and a Haling calculation.

The three different options all require the beginning of cycle region burnups. These are determined by input or by the shuffle from the previous cycles. The end of cycle burnups from the previous cycles are modified, if necessary, to reflect partial discharge as discussed in Section 2.8 and used as the beginning of cycle burnups for the current cycle.

The straight depletion options start with the beginning of cycle burnups, calculate relative power sharings and then incrementing the region burnups by the burnup step size times the relative region power sharings. The nominal step size in the program is set at 4 GWD/MTU with a shorter time step taken last to give the exact desired cycle lifetime. A large time step is used since the variation of power sharings with burnup in a cycle is usually small on a region-wise basis. Therefore, the error is small.

For the no control depletion, the interpolated k_∞ are used directly without assuming any form of control. With uniform control, a search is performed each depletion step to determine the uniform change in k_∞ to be applied to each region to yield the desired k_{eff} of 1.0 within $\pm$ .00001. The uniform control option is used when the boron worth is input.

With a Haling (6) calculation option, an iteration is performed to give the end of cycle power distribution which is held constant for the entire cycle depletion. Thus an initial power distribution is assumed; the EOC region-wise burnups are then determined using these power sharings; and then the EOC power distribution is calculated. This new power distribution is then used to recalculate the region-wise EOC burnups. This procedure is used iteratively until the maximum region-wise power sharing difference between iterations is less than 0.001.

Thus, the Haling option assumes that the reactor will be controlled so that a constant power shape will be maintained during the core life. Since end of life implies that there is no excess reactivity; there is no control present. Therefore, the Haling calculation assumes that the power distribution during the cycle is identical to the end of cycle power distribution which has no control.

2.8 Discharge Burnups

The number of assemblies discharged in any cycle is equal to the total number of assemblies in the region minus the total number of assemblies used from that region in all future cycles. It is assumed that the burnup of all assemblies used in future cycles are identical and that the discharge burnup of each assembly discharged is identical. But an option exists to allow for different burnups between discharged and kept assemblies. It is assumed that the distribution of burnups in a region is uniform over a specified range. Also, it is assumed that the highest average burnup assemblies will be discharged and the lowest burnup assemblies will be kept. If there is a uniform distribution of assembly burnups within a range of $Bu_{avg} \pm \Delta Bu$, then the burnups are given by:

$$Bu^{Discharged} = Bu_{avg} + \Delta Bu \left(1 - \frac{\#\ Discharged}{\#\ Region}\right) \qquad (2.29)$$

$$Bu^{Kept} = Bu \qquad - \Delta Bu \left(\frac{\#\ Discharged}{\#\ Region}\right) \qquad (2.30)$$

where the numbers are the number of assemblies discharged or in the region. A typical value of ΔBu is 2 GWD/MTU.

2.9 Burnable Poison Treatment

The burnable poisons are treated on a region-wise basis. For each cycle the total number of BPs are given and the fraction in each region.

This allows for a search on the number of BP rods in the whole core if desired. The depletion of the BP rods is handled on a functional basis of the region average burnup. Thus the BP rods are not actually depleted. The worth in one assembly of a single BP rod is tabulated as a function of region burnup. The code assumes a linear relationship between worth and the number of BP rods. This is a good assumption for a small number of BP rods/assembly. Therefore, to determine the change in k_{eff} for a region due to burnable poisons, the following relationship is used:

$$k_{eff}(\text{w BP}) = k_{eff}(\text{wo BP}) - \frac{(\text{No. BPs})(\text{Fraction in Region})}{\text{No. Assemblies in Region}} \rho_{BP}(\text{Bu}) \tag{2.31}$$

where $\rho_{BP}(\text{Bu})$ is the worth of one BP rod at a burnup Bu. Note that the BP worth is determined at the region average burnup, so there is an implicit function that fresh BPs are only placed in fresh fuel.

2.10 Cycle Search Procedure

It is possible to search for any one of a number of variables each cycle. The allowed search variables are:

cycle burnup

number of burnable poison rods

EOC k_{eff}

enrichment of any feed region

These searches are all performed using a Newton-Raphson procedure. The new value of the variable y for iteration i+1 is given by:

$$y^{i+1} = y^{i} - \text{slope}\,(k_{eff}^{i} - k_{eff}^{want}) \tag{2.32}$$

$$\text{slope} = \frac{y^{i} - y^{i-1}}{k_{eff}^{i} - k_{eff}^{i-1}}$$

Convergence is assumed when $|y^{i+1} - y^{i}| < \varepsilon$ where ε is .001 for cycle burnups, 1.0 for number of BP rods, and .01 for enrichment.

These convergence criteria were set to yield acceptable results without unnecessary accuracy. They also allow for significant changes in the k_{eff} to insure that division by the difference in k_{eff} does not produce overflow problems in the computer. Note that no iteration is needed for the EOC k_{eff}, this is just a straight forward calculation. To eliminate possibly large variations in feed enrichment, allowable minimum and maximum values of enrichment are input. When the enrichment hits the edge of this band, it is set to the value and the k_{eff} calculated.

2.11 Convergence Criteria

The program uses many different iterative scheme to calculate the desired result. In some cases there can be a significant nesting of iterative searches. Up to five iterations nestled together is possible. These different iterative schemes are located in three different subroutines and have built in convergence criteria. The various convergence criteria are listed in Table 2.1 along with the subroutine where it may be found.

TABLE 2.1

CONVERGENCE CRITERIA

Routine	Parameters	Max. Iterations	Epsilon
EQCALC	EOC burnups	50	0.01
CYCALC EQCALC	Cycle burnup	50	0.001
CYCALC EQCALC	Number of BP rods	50	1.0
CYCALC EQCALC	Feed enrichment	50	0.001
CYCALC	Boron defect	-	0.0001
CYCALC	EOC Haling burnups	50	0.001
SOLVE	Eigenvector Eigenvalue	50	0.001 0.00001

CHAPTER 3

VERIFICATION AND QUALIFICATION

The checkout of a computer program consists of two separate and distinct parts - verification and qualification. The verification of a program consists of demonstrating that the program performs the calculations as specified. The qualification of a program consists of determining the applicability of the program for its use to calculate various parameters using a prescribed method. Thus verification consists of demonstrating that the program does what it is supposed to do and qualification consists of showing that what it does gives good results using certain methods.

Sensitivity studies were also performed to determine how accurate the input needs to be to give acceptable results. These studies demonstrate the adequacy of one value for the albedo and the absorption probability (w) and the adequacy of the automatic generation of the edge data.

3.1 Verification

There are a number of areas which can be separately verified. The two basic areas are the actual solution of the matrix equation and the interpolation of data. These two aspects form the basis of the program. All the other aspects can be verified by successful execution of the program. For example, the various search procedures must converge to a correct solution or be wrong.

3.1.1 Matrix Verification

A 3 x 3 matrix was chosen for verification since it can readily be solved by hand. For a 3 x 3 matrix given by

$$A = \begin{bmatrix} \beta_1 & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \beta_2 & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \beta_3 \end{bmatrix} \tag{3.1}$$

The eigenvalues are found by solving the equation det $(A - \lambda I) = 0$. Writing out the determinant gives:

$$\lambda^3 - (\beta_1 + \beta_2 + \beta_3)\lambda^2 + (\beta_1\beta_2 + \beta_1\beta_3 + \beta_2\beta_3 - \gamma_{13}\gamma_{31} - \gamma_{23}\gamma_{32} - \gamma_{12}\gamma_{21})\lambda$$
$$+ (\beta_1\gamma_{23}\gamma_{32} + \beta_2\gamma_{13}\gamma_{31} + \beta_3\gamma_{12}\gamma_{21} - \beta_1\beta_2\beta_3 - \gamma_{12}\gamma_{23}\gamma_{31} - \gamma_{13}\gamma_{21}\gamma_{32})$$
$$= 0 \tag{3.2}$$

The largest eigenvalue is found by solving the above third order polynomial. This can be done easily by an iterative search. Given the eigenvalue, the eigenvector is then calculated by:

$$S_1 = 1.0$$
$$S_2 = \frac{\lambda - \beta_1}{\gamma_{12}} - \left(\frac{\gamma_{13}}{\gamma_{12}}\right) S_3 \tag{3.3}$$
$$S_3 = \left[\frac{\lambda - \beta_1}{\gamma_{12}} - \frac{\gamma_{21}}{\lambda - \beta_2}\right] \Big/ \left[\frac{\gamma_{23}}{\lambda - \beta_2} + \frac{\gamma_{13}}{\gamma_{12}}\right]$$

A sample case was set up to test the matrix solution procedure. The problem was a three region problem as specified in Table 3-1. The values of α and w used are:

$$w = 0.10 \text{ and } \alpha = 0.30$$

The corresponding matrix for this problem as defined in Equations 2.9, 2.10 and 2.12 is:

$$\begin{pmatrix} .60234 & .34855 & .053005 \\ .389791 & .66066 & .050650 \\ .066852 & .056948 & .92880 \end{pmatrix}$$

The cubic polynomial giving the eigenvalues of this matrix using Equation 3.2 is:

$$\lambda^3 - 2.191797\lambda^2 + 1.428722\lambda - 0.2416968 = 0 \qquad (3.4)$$

which gives as a solution $\lambda = 1.0527$ within the accuracy of a hand calculator. Applying Equation 3.3 gives the corresponding eigenvector:

$$\underline{S} = \begin{pmatrix} 1.0 \\ 1.131 \\ 1.0589 \end{pmatrix}$$

TABLE 3.1

MATRIX SOLUTION TEST PROBLEM

Region	k_{eff}	Fraction of Edges Adjacent to Region			
		1	2	3	Exterior
1	1.0039	0.0	0.8680	0.1320	0.0
2	1.1011	0.8850	0.0	0.1150	0.0
3	1.2380	0.1350	0.1150	0.4620	0.2880

The answer calculated by the program was

$$\lambda^* = 1.0527 \text{ and } \underline{S}^* = \begin{pmatrix} .940 \\ 1.06 \\ .999 \end{pmatrix}$$

where $\lambda^* = \lambda$ and $\underline{S}^* = .940\ \underline{S}$.

Note that since S*, S are eigenvectors they can be normalized in any fashion. Hence the fact that S* = 0.94S implies equality within a multiple and hence implies that S and S* are identical eigenvectors. Therefore, the program is correctly computing the matrix and its largest eigenvalue and its corresponding eigenvector.

3.1.2 Interpolation Verification

The interpolation scheme was also verified. From the data in Table 3.2, the k_∞ for 3.20 w/o at a burnup of 22.0 GWD/MTU was calculated. The program interpolated value was 1.1011.

As described in Section 2.6, the interpolation is done at the specified burnup for each enrichment and then the value for the specified enrichment is calculated. From equation (2.27) we get the relationship:

$$k_\infty(22) = -.083333k_\infty(8) + 1.020833k_\infty(20) + .0625k_\infty(36) \qquad (3.5)$$

so

$$k_\infty^{2.9\ w/o}(22) = 1.07506$$

$$k_\infty^{3.1\ w/o}(22) = 1.09243$$

TABLE 3.2

DATA FOR k_∞ INTERPOLATION

Burnup (GWD/MTU)	k_∞ (2.90 w/o)	k_∞ (3.10 w/o)
1	1.31136	1.32605
8	1.22296	1.23898
20	1.09447	1.11178
36	.95529	.97183

From Equation (2.28) we also have that

$$f^{3.2}(22) = -0.5\ f^{2.9}(22) + 1.5\ f^{3.1}(22)$$
$$f^{3.2}(22) = 1.1011 \qquad (3.6)$$

Thus the hand calculation verifies the computer calculation for the interpolation and extrapolation.

3.2 Qualification

The verification demonstrates that the FLAC program is functioning as formulated. It remains to be demonstrated that the program is qualified for use in fuel management studies. In order to evaluate the effectiveness of a computer program, a large number of varied cases should be run. The work done by Rieck (10) in his doctoral thesis is well suited to provide benchmark cases for the FLAC code qualification. A large number of variations in the number of feed assemblies for the Zion reactor was investigated. Zion is a 4 loop PWR reactor by Westinghouse. Its core consists of 193 assemblies distributed in 15 rows. The information needed to generate the assembly k_∞ data consistently with Rieck's data is available in his thesis. Based on current knowledge, I believe that the data presented by Rieck is not in perfect agreement with current calculations for the Zion core. However, the use of the same k_∞ data assures that the comparison is meaningful. A comparison using the same basic input as was used with more precise methods demonstrates how closely FLAC reproduces the more precise calculations. As with any computer code, the accuracy of the input will determine the accuracy of the output to a large degree. What the qualification demonstrates is the ability of FLAC to obtain essentially the same results as a more precise tool using the same basic input.

Two depletion cases were run for eight cycles - one feeding 64 assemblies at 3.20 w/o and the other feeding 48 assemblies at 3.76 w/o. Then various perturbation calculations were performed for cycle 9 with variations in the number of feed assemblies from 48 to 84 and significant variations in enrichment. There were 19 change cases starting from the 64 feed assembly case and 6 change cases starting from the 48 feed assembly case.

The model employed in FLAC used the k_∞ as generated from the SIMULATE fit as used by Rieck. The automated generation of edge data with a checkerboard interior and the automated shuffling option were used. The number of feed assemblies and the feed enrichment were input and the program was allowed to calculate the cycle lifetime. A boron search was performed and a discharge burnup variation parameter of 2.0 GWD/MTM was used. The desired end of life k_{eff} was 1.0001 which corresponds to approximately 10 ppm of residual boron which would translate into about 0.01 GWD/MTU. The results of the comparison are presented in Tables 3.3 through 3.6.

As can be seen from comparing the results, the FLAC calculations agree very well with the more detailed calculations performed by Rieck. The results for all 39 cycles yield an average difference in the cycle burnup of only 79 MWD/MTU (less than 1% error) and a standard deviation of 104 MWD/MTU. This is excellent accuracy for a survey tool since the cycle average burnup is seldom known by even the best methods to within $\pm$ 100 MWD/MTM. This is due to the fact that there does exist manufacturing, design, enrichment and cross section uncertainties all of which create uncertainty in the final design calculations.

TABLE 3.3

ZION 64 FEED ASSEMBLY CYCLE STUDY

Cycle	3-D Calculations Cycle Burnup (GWD/MTU)	3-D Calculations Discharge Burnup (GWD/MTU)	FLAC Cycle Burnup (GWD/MTU)	FLAC Discharge Burnup (GWD/MTU)	Difference Cycle Burnup (GWD/MTU)	Difference Discharge Burnup (GWD/MTU)
1	15.600	16.943	*	17.407	-	.207
2	9.652	25.115	9.715	25.800	.063	.685
3	9.894	32.076	10.095	31.672	.201	-.404
4	10.284	30.306	10.164	30.331	-.120	.025
5	10.038	30.401	10.050	30.502	.012	.101
6	10.084	30.419	10.104	30.471	.020	.052
7	10.081	30.399	10.093	30.456	.012	.057
8	10.081	30.400	10.090	30.408	.009	.008
			Average		.028	.091
			Standard Deviation		.088	.280

*Because of burnable poisons, the search was on the calculated k_{eff} at end of life.

TABLE 3.4

ZION 48 FEED ASSEMBLY CYCLE STUDY

Cycle	3-D Calculations Cycle Burnup (GWD/MTU)	3-D Calculations Discharge Burnup (GWD/MTU)	FLAC Cycle Burnup (GWD/MTU)	FLAC Discharge Burnup (GWD/MTU)	Difference Cycle Burnup (GWD/MTU)	Difference Discharge Burnup (GWD/MTU)
1	15.600	17.174	*	17.899	-	.725
2	8.228	23.917	8.436	25.394	.208	1.477
3	8.887	31.559	9.419	32.224	.532	.665
4	9.868	37.029	9.780	38.239	-.088	1.210
5	10.151	38.433	10.422	38.897	.271	.464
6	9.659	39.191	9.731	39.710	.072	.519
7	9.769	39.488	9.892	40.092	.123	.604
8	9.811	39.370	9.943	39.892	.132	.526
	Average				.179	.774
	Standard Deviation				.178	.344

*Because of burnable poisons, the search was on the calculated k_{eff} at end of life

TABLE 3.5

ZION VARIATION FROM 64 FEED ASSEMBLIES

Case	Number	Enrichment (w/o U-235)	3-D Burnup (GWD/MTU)	FLAC Burnup (GWD/MTU)	Difference (GWD/MTU)
1	56	4.42	10.953	11.049	.096
2	60	3.86	10.709	10.785	.076
3	64	3.70	11.057	11.138	.081
4	68	3.40	11.007	11.063	.056
5	72	3.30	11.279	11.321	.042
6	76	3.06	11.104	11.146	.042
7	80	2.94	11.174	11.211	.037
8	60	4.34	11.606	11.666	.060
9	64	3.96	11.618	11.639	.021
10	68	3.78	11.874	11.873	-.001
11	64	4.20	12.097	12.123	.026
12	68	3.88	12.088	12.086	-.002
13	76	3.42	12.103	12.106	.003
14	68	4.34	13.100	13.115	.015
15	72	4.00	13.019	12.955	-.064
16	80	3.60	13.118	13.087	-.031
17	72	4.50	14.127	14.225	.098
18	76	4.20	14.131	14.096	-.035
19	84	3.76	14.150	14.076	-.074
		Average			+25.6
		Standard Deviation			48.6

TABLE 3.6

ZION VARIATION FROM 48 FEED ASSEMBLIES

Case	Number Fed	Enrichment (w/o U-235)	3-D Burnup (GWD/MTU)	FLAC Burnup (GWD/MTU)	Difference (GWD/MTU)
1	56	4.06	11.780	11.918	.138
2	60	3.74	11.861	12.007	.146
3	68	3.30	12.023	12.230	.207
4	64	4.38	13.823	13.950	.127
5	68	4.06	13.850	13.914	.064
6	76	3.52	13.822	13.826	.004
				Average	114.3
				Standard Deviation	64.6

These cases covered a very broad range of conditions going from 48 feed assemblies to 84 feed assemblies in a variety of loading patterns. These cases also covered successive cycles and demonstrated that the deviations do not become larger cummulatively. Instead they demonstrate convergence as the equilibrium cycling scheme is approached. Yet the edge data was generated in FLAC automatically for each case, thus demonstrating the adequacy of this procedure for predicting average cycle burnups.

A comparison of the region discharge burnups in Tables 3.3 and 3.4 also shows good agreement. As expected, the average discharge burnup variation in the three region core is approximately three times the cycle burnup variation and the average discharge burnup variation in the four region core is approximately four times the cycle burnup variation. The largest discrepancies appear to be associated with regions 1, 2 and 3 which were all in cycle 1 with burnable poisons. Very little work was done with the burnable poison treatment and it is expected that future work could refine this calculation. Also, there are large discrepancies in the four region case for cycles 2 through 4 where partial regions were discharged. One primary reason for these differences is the method of selecting the assemblies to be discharged. FLAC seeks to optimize fuel utilization and hence discharges the highest burnup fraction of the region. In the comparison study the selection seems to be fairly random with in some cases the lowest burnup assembly in a region being discharged.

The actual power or burnup sharings between regions in a cycle is subject to more variation since it is highly dependent on the loading pattern. Table 3.7 shows the comparison between the actual burnup sharings from the explicit multi-dimensional calculation and the FLAC calculation for an equilibrium cycle in Zion with both 48 and 64 feed assemblies. As can be

TABLE 3.7

BURNUP SHARING COMPARISON

64 Feed Assemblies

Region	3-D Burnup Sharing (GWD/MTU)	FLAC Burnup Sharing (GWD/MTU)	Number of Assemblies
1	8.2	8.2	1
2	8.3	8.7	64
3	10.8	10.4	64
4	10.5	10.7	64

48 Feed Assemblies

Region	3-D Burnup Sharing (GWD/MTU)	FLAC Burnup Sharing (GWD/MTU)	Number of Assemblies
1	7.5	6.8	1
2	7.9	8.6	48
3	10.1	8.9	48
4	12.3	11.8	48
5	9.9	10.0	48

seen the results are generally within $\pm$ 10% and for the 64 feed assembly, the results are excellent.

Based on these comparisons, FLAC appears to generate acceptable results for long range fuel management studies. The cycle length will be estimated within $\pm$ 500 MWD/MTU with a high degree of certainty assuming the input is generated properly.

3.3 Sensitivity Analysis

FLAC is an approximate method. It is not a detailed and sophisticated procedure to yield very accurate results. Therefore, it does not require extremely accurate input data. Some of the input data does vary somewhat with particular core configurations (i.e., W and ALPHA, the absorption probability and the albedo), yet only average values are used. Other parameters will vary depending on the actual loading pattern (number of edges), yet only a general type of loading pattern is assumed. This section will discuss the various analyses which were done to quantify the sensitivity to these variables.

3.3.1 Sensitivity to W and ALPHA

A 193 assembly core, 3 region case was calculated for different combinations of values of W, the probability that a neutron born in one node will be absorbed in an adjacent node, and of ALPHA, the albedo. The range of values was W from 0.100 to 0.140 and ALPHA from 0.25 to 0.50. The values that have been found to give the best results for many different cases are W equal to 0.115 and ALPHA equal to 0.30. The sensitivity analysis demonstrates that variation in the parameters by $\pm$ 10% yields results with acceptable accuracy ($\pm$ .005 $\Delta\rho$ and $\pm$ 5% in regionwise power). These results are graphically displayed in Figures 3.1 through 3.8. These figures indicate that the inner regions are relatively insensitive to the variations. Thus the major variations are the peripheral region power sharing and the k_{eff}.

3.3.2 Sensitivity to Edge Data

Cycle length and power sharings are affected by the loading pattern choice, yet when running FLAC one does not want to have to know the precise loading pattern in each cycle. One would like the results to be relatively insensitive to the loading pattern, while being capable of determining the relative merits of different types of loading pattern schemes. And it appears that FLAC can do this.

First, the results are relatively independent of the variations in a loading pattern within a particular scheme. All of the verification runs used the automatic checkerboard loading edge data routine built into the code. Thus the verification included random deviations from the actual loading pattern assuming the general constraint of a peripheral feed with a checkerboard interior.

A separate study was performed with edge data corresponding to various loading patterns. The data corresponds to the actual loading pattern in a three region core using a checkerboard interior and automatically generated for a checkerboard pattern, a ring pattern and checkerboard patterns with burnt fuel in the periphery. The results are summarized in Table 3.8. As can be seen from this table, there are only minor differences between the actual checkerboard pattern and the code generated one, yet significant differences can be noticed between the other loading patterns. For example, the cases with the burnt assemblies

Figure 3-1

Sensitivity Study, k_{eff} versus W and α

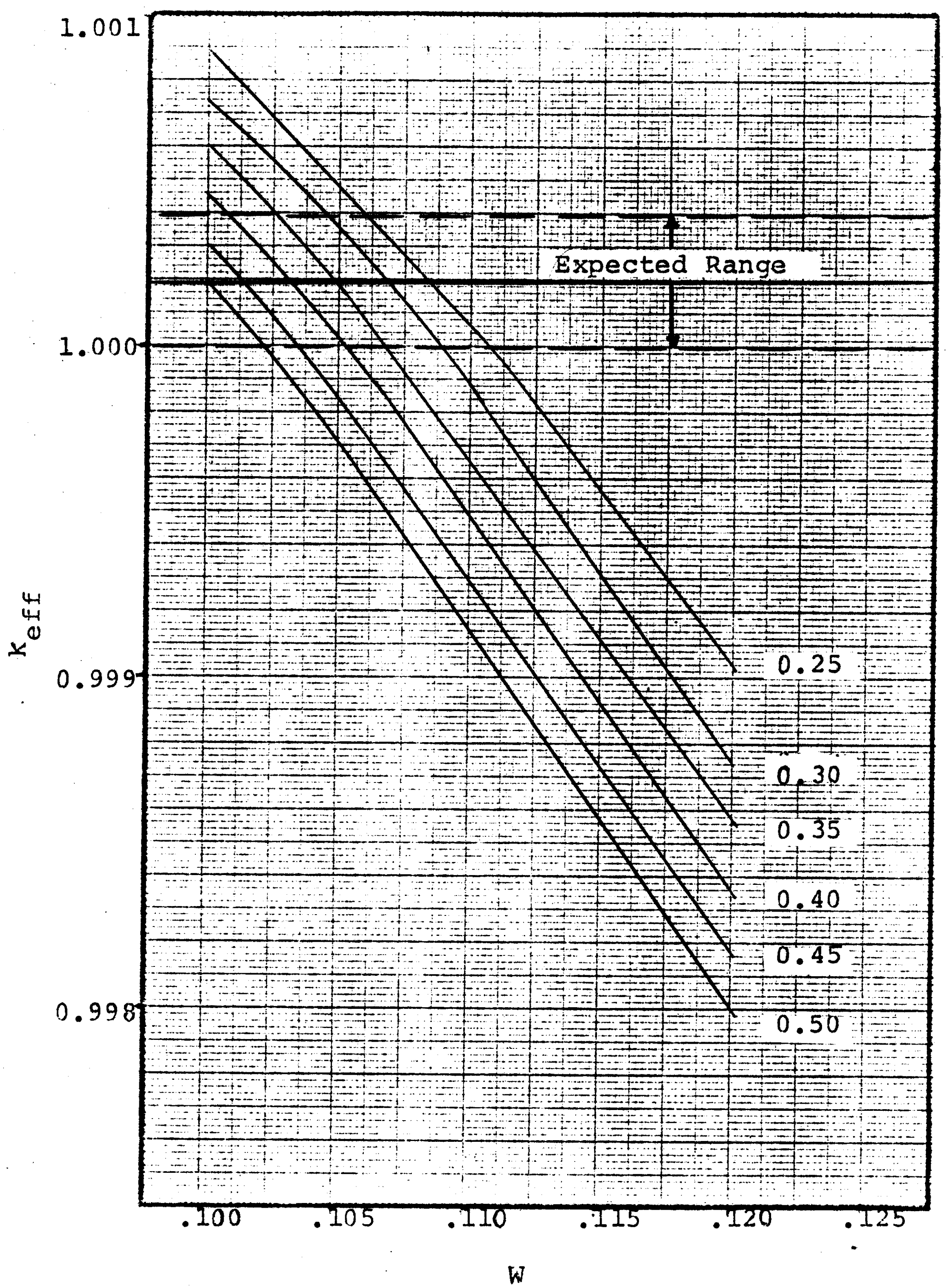


Figure 3-2

Sensitivity Study, k_{eff} versus W and α

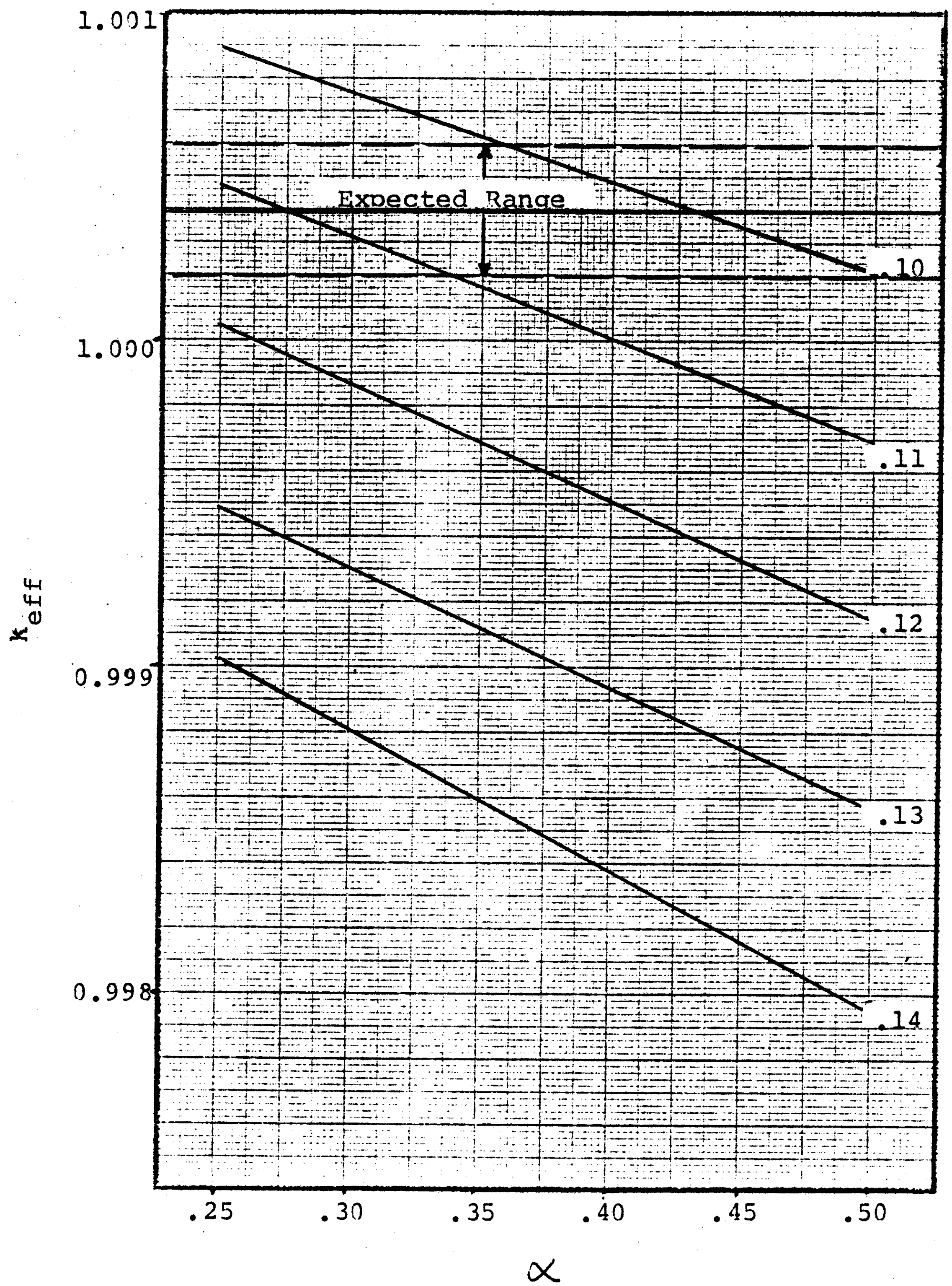


Figure 3-3

Sensitivity Study, Region 1 Power versus W and α

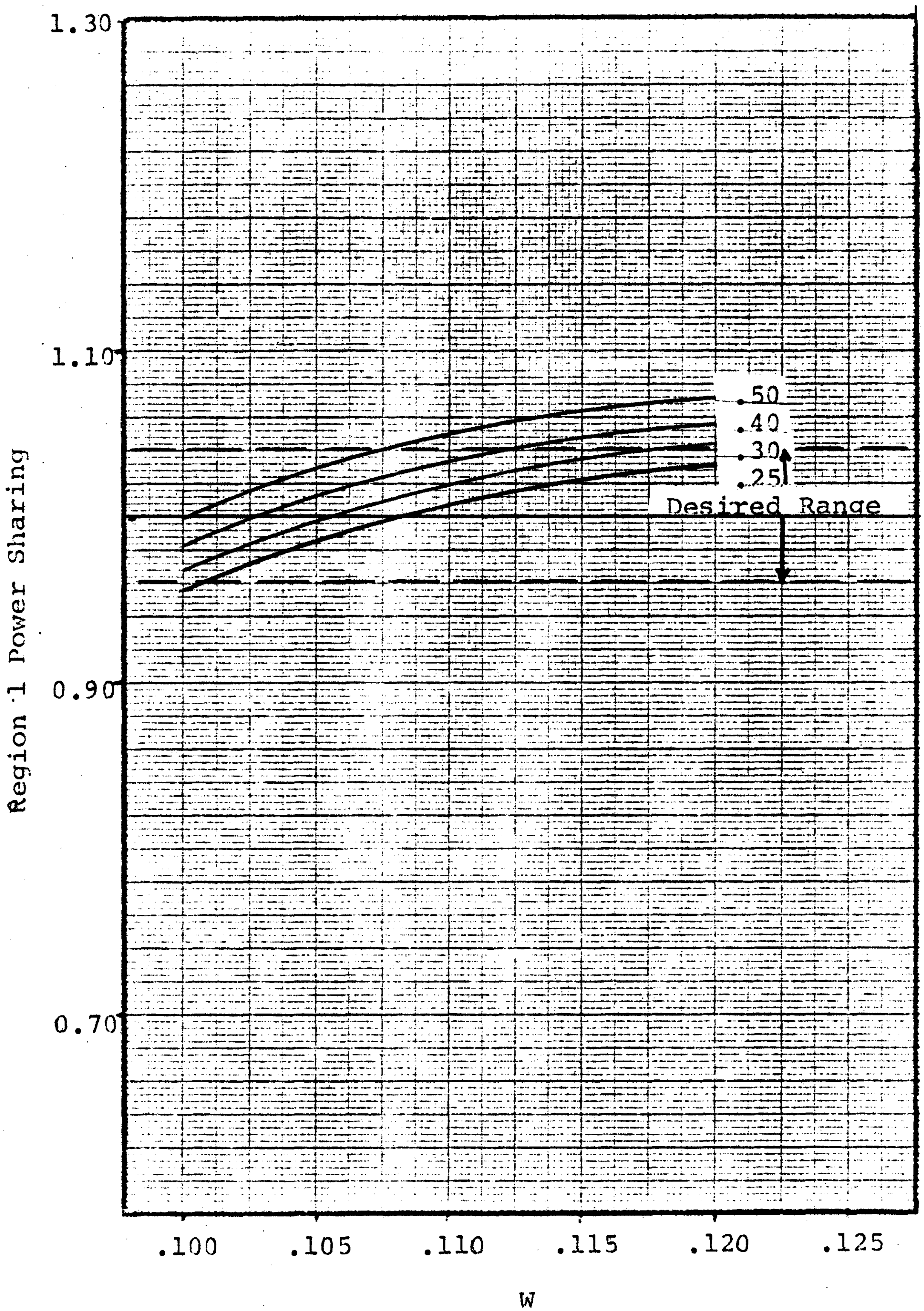


Figure 3-4

Sensitivity Study, Region 1 Power versus W and α

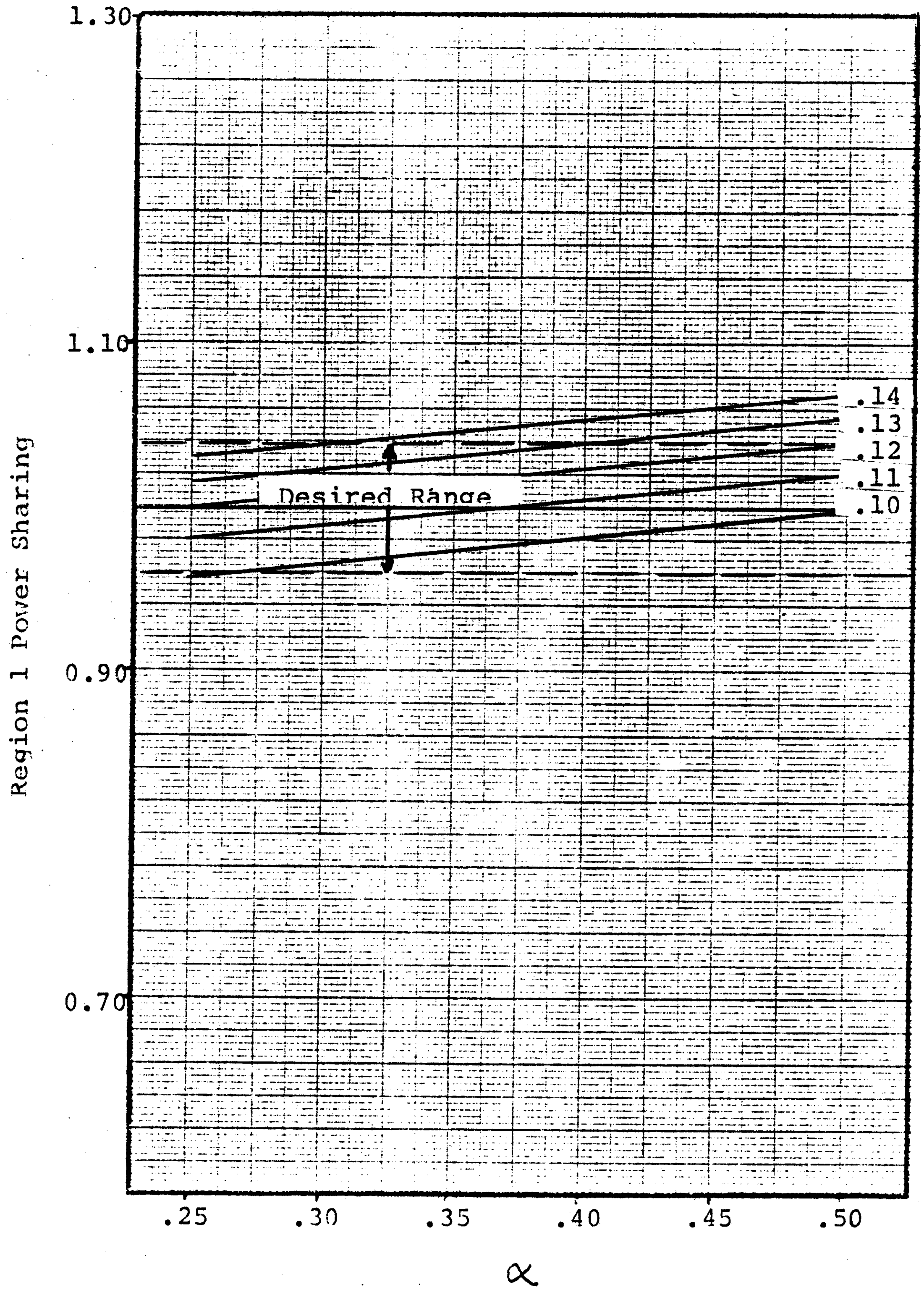


Figure 3-5

Sensitivity Study, Region 2 Power versus W and ∝

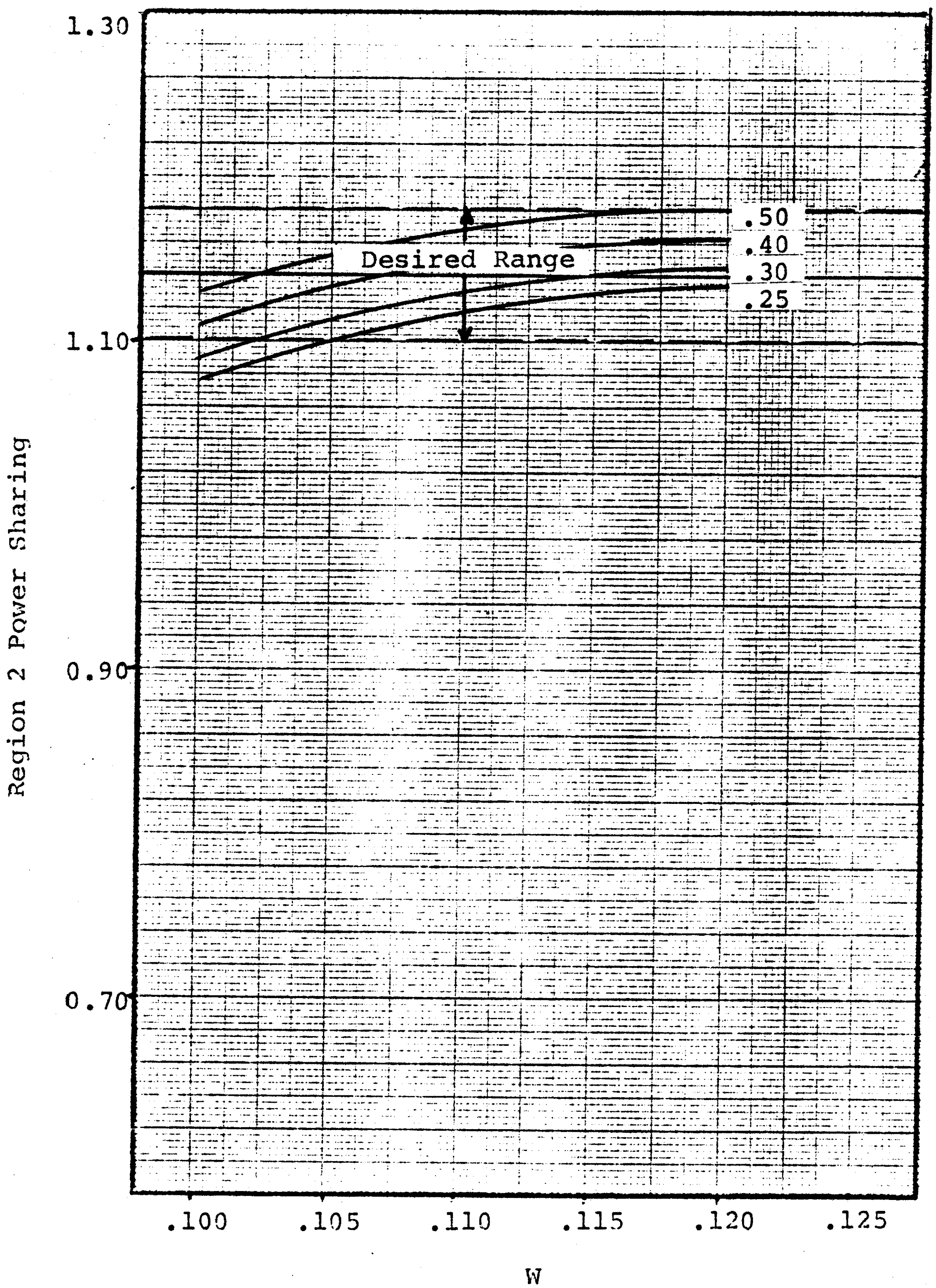


Figure 3-6

Sensitivity Study, Region 2 Power versus W and α

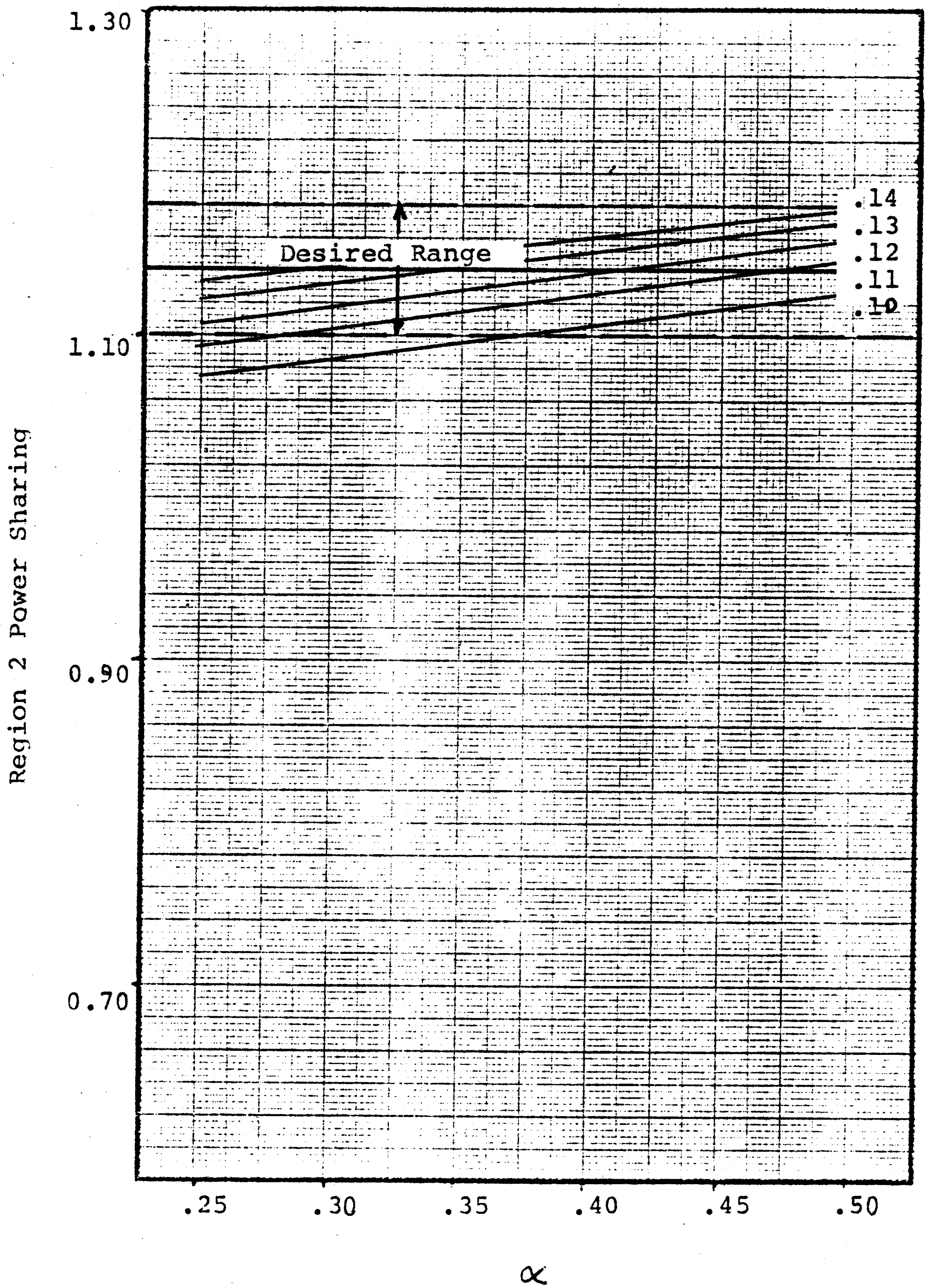


Figure 3-7

Sensitivity Study, Region 3 Power versus W and α

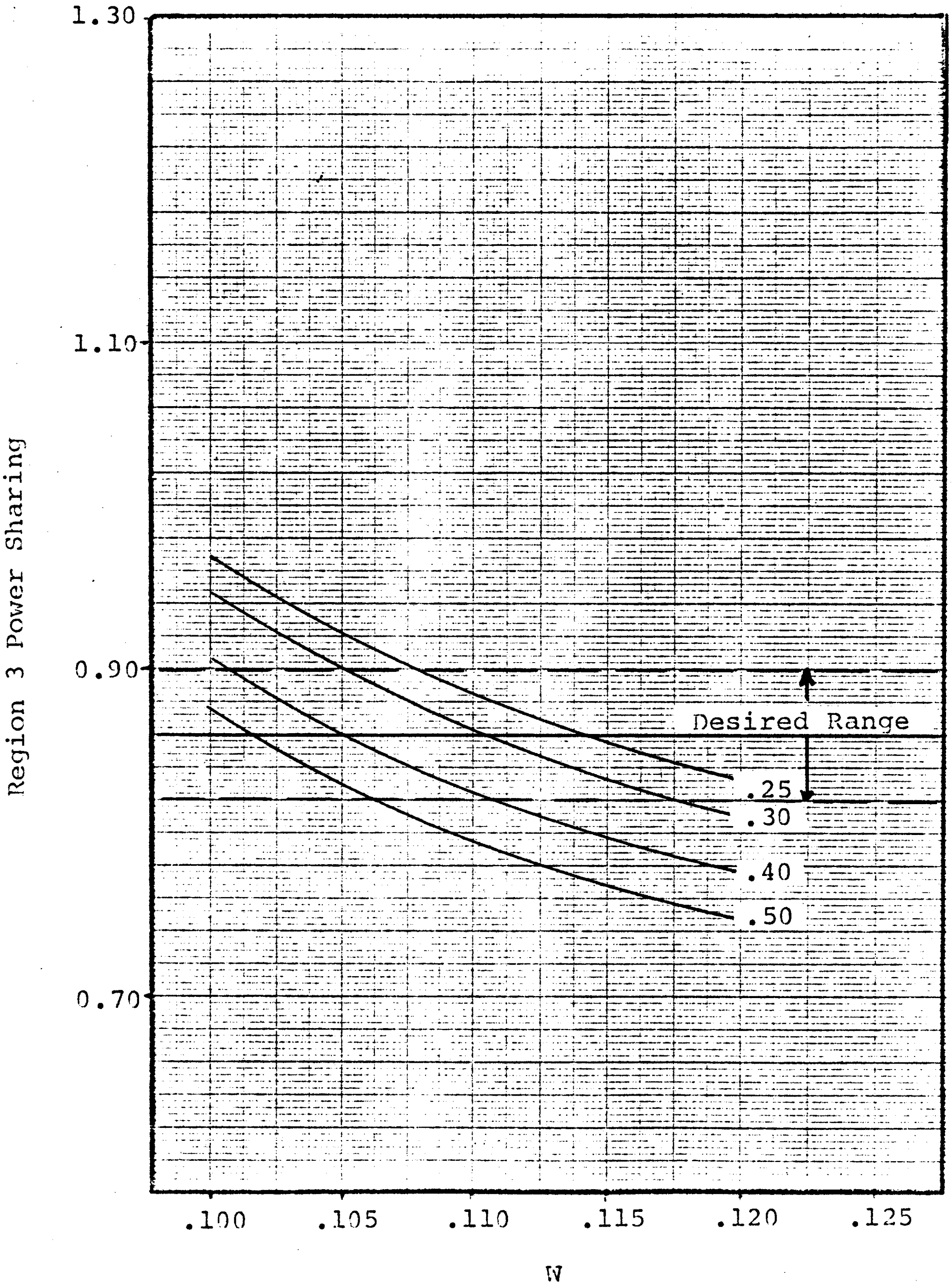


Figure 3-8

Sensitivity Study, Region 3 Power versus W and α

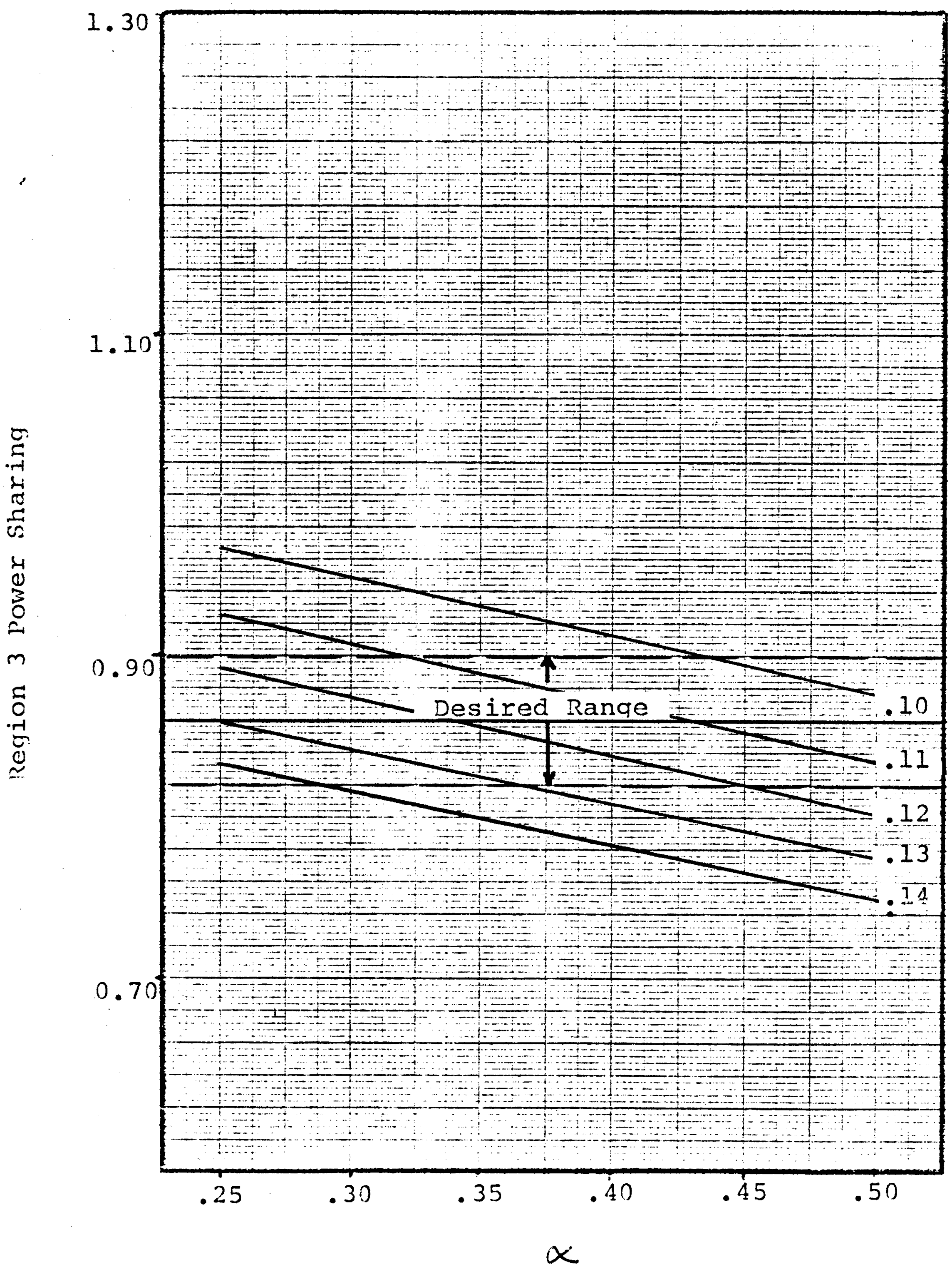


TABLE 3.8

LOADING PATTERN SENSITIVITIES

		Actual Checkerboard	Automatic Checkerboard	Automatic Ring	Checkerboard 12 Burnt on Periphery	Checkerboard 24 Burnt on Periphery
Cycle Length (GWD/MTU)		11.01	10.98	10.40	11.09	11.62
Power Sharings						
Reg 3	BOL	.933	.943	1.122	1.154	1.284
	EOL	.900	.905	.953	1.063	1.176
Reg 2	BOL	1.110	1.106	1.262	1.058	1.047
	EOL	1.108	1.106	1.169	1.086	1.091
Reg 3	BOL	.961	.954	.626	.792	.674
	EOL	.994	.991	.883	.852	.736

Based on: 157 Assembly Core
52 Feed Assemblies at 3.20 w/o U-235
Equilibrium Cycle

on the periphery show increased cycle lifetime due to reduced neutron leakage. However, these cases also indicate higher powers in the feed region. The ring model agrees with the general view that the checkerboard patterns are superior to ring patterns for larger cores since rings have higher peaking and less cycle lifetime.

A second study was done using the Zion model as discussed in Section 3.2. Equilibrium cycle calculations were done to determine the feed enrichment required for a fixed 11.0 GWD/MTM cycle length for various numbers of feed assemblies from 48 to 84. These cases were run with a checkerboard interior and a ring interior. These results are summarized in Table 3.9 and the required enrichment given in Figure 3.9. These results also demonstrate the superiority of the checkerboard pattern. It also indicates the variation in power sharings as the number of feed assemblies is changed. This type of data is presently impossible to obtain without explicit multi-dimensional calculations.

Therefore, FLAC presents a good compromise on the need for accurate knowledge of a loading pattern. It can differentiate between different basic types but does not need a detailed description of the loading pattern.

3.3.3 Sensitivity to Boron Search

An option exists in FLAC to calculate a uniform poison worth each step and use this to modify the region k_∞'s. This is comparable to a search to criticality using the soluble boron concentration as the search variable. Two calculations, one with and one without the search, were performed to determine the effect of the search. The results are listed in Table 3.10. These show that the effect is fairly small with the largest effect at

TABLE 3.9

EQUILIBRIUM FEED ENRICHMENT

Number of Feed Assemblies	Checkerboard Pattern		Ring Pattern	
	Feed Enrichment (w/o U-235)	Peak Power	Feed Enrichment (w/o U-235)	Peak Power
48	4.133	1.207	4.424	1.651
52	3.943	1.165	4.216	1.477
56	3.822	1.308	3.987	1.481
60	3.615	1.216	3.780	1.407
64	3.443	1.161	3.619	1.391
68	3.339	1.148	3.503	1.402
72	3.243	1.147	3.367	1.343
76	3.156	1.141	3.246	1.311
80	2.070	1.123	3.145	1.289
84	2.987	1.097	3.052	1.284

Based on: 193 Assembly Core
11.0 GWD/MTM Cycle Length
Equilibrium Cycle

Figure 3-9

Equilibrium Feed Enrichment

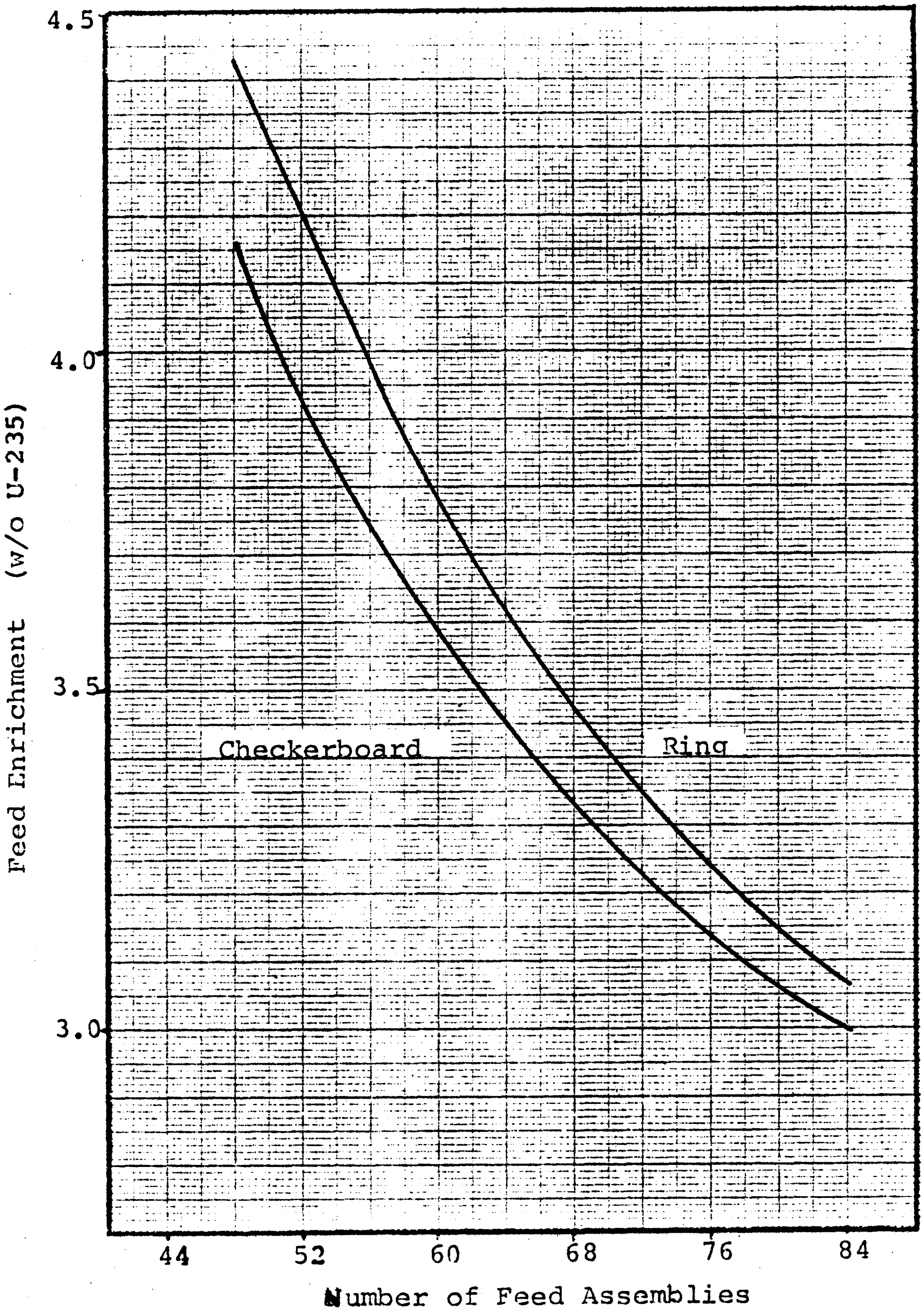


TABLE 3.10

SENSITIVITY TO BORON SEARCH

Parameter		Boron Search	No. Search
EOL k_{eff}		0.9992	1.0002
Power Sharings			
Region 3	BOL	.908	.882
	EOL	.886	.887
Region 2	BOL	1.121	1.128
	EOL	1.114	1.114
Region 1	BOL	.975	.993
	EOL	1.002	1.001

Based on: 157 assembly core
52 Feed Assemblies at 3.2 w/o U-235
Cycle length of 11.00 GWD/MTU
Equilibrium Cycle

beginning of life where the poison concentration is the highest. However, at end of life the effect is very small. The effect of the boron search is within the uncertainty of the calculation and its use is optional. However, to assure consistant comparison of different cases; the same option should be used in each case.

CHAPTER 4

CONCLUSIONS AND RECOMMENDATIONS

4.1 Recommended Method

FLAC is an approximate tool giving results only within a specified range. Care must be taken to avoid losing sight of this basic characteristic. When preparing the input to FLAC, therefore, one should not get buried in details which in the end will have no significant affect on the results.

The interpolation scheme in FLAC gives very good results over fairly wide ranges of enrichment. Therefore, I would recommend enrichments spaced at intervals of about 0.5 w/o U-235. In fact, data corresponding to two enrichments very close in enrichment can cause errors since there are usually secondary parameters which vary from region to region in addition to the enrichment. And for very small changes in enrichment, these secondary changes could be quite significant. As far as burnup variations, steps of 4.0 GWD/MTM are generally sufficient and even larger steps would probably be adequate.

The actual k_∞ data can be taken from LEOPARD (2) calculations which properly homogenize the whole assembly. The k_∞ should correspond to full power fuel temperatures and moderator densities, contain equilibrium xenon, samarium and the other fission products and <u>not</u> contain any soluble boron.

The isotopic information can be generated using the built in fit unless significant variation is seen or unless a different fuel type (i.e. thorium) is used.

The use of the automated shuffle option and the automated edge data generation option are highly recommended. There will probably be more cases to explicitly input the shuffle than the edge data. For example, when assemblies are reinserted into the core after sitting out one or more cycles or when different shuffling schemes are to be tried which do not have the feed region exclusively in the periphery. However, since most loading patterns are variations of a checkerboard loading pattern, this option is highly recommended.

The use of the boron search option feature is optional since its effect is small. However, not using this option does slightly simplify the the input.

Since the results are relatively insensitive to the values of W and ALPHA, it is recommended that the default values be used for standard PWR cores. For BWR cores or non-standard cores, a study should be performed to determine the optimum values of these parameters.

For the search procedure, it is recommended that the burnup search be used most frequently. The enrichment search should be used mainly in equilibrium cycle calculations where large fluctuations in enrichment will not occur.

The one big assumption in generating the FLAC equations from the FLARE equations is the homogeneity of the region. It assumed that the k_∞ is identical for each assembly in the region. This assumes that the burnup and enrichment are the same for each assembly within a region. This is usually not possible, but regions as used by the code should be assemblies having the same enrichment and approximately the same burnup at beginning and end of life. Thus split enrichment feeds should be divided up as well as regions placed both on the periphery and in the interior.

The use of the above methods should produce good results with a minimum of effort. Their use is therefore recommended.

4.2 Method Accuracy

Based on the verification, qualification and sensitivity analysis performed, the recommended method will produce good results. In long range fuel management there are many variables which can alter the results of a study. Many of these variables are external and can not be controlled. Therefore, it is superfulous to calculate any of the parameters to a high degree of accuracy.

The FLAC program using the recommended procedure can predict the cycle length to within ± 500 MWD/MTM to a high degree of confidence. Likewise, it can predict the region power sharings to within ± 10%. A higher degree of accuracy is possible when parameter variations are made for comparisons of their relative effects. This degree of accuracy is sufficient for almost all long range fuel management studies.

4.3 Conclusions

A new long range fuel management program, FLAC, has been developed and its use demonstrated. The program has been written, verified and qualified for fuel management studies. The program can differentiate between the various types of loading patterns on a generic basis and thus is a significant improvement over the weighted k procedures which have been used in the past. However, variations within loading pattern types have been shown not to be significant, hence explicit loading patterns are not required which simplifies the fuel management study considerably compared to multi-dimensional depletion calculations. Therefore, FLAC satisfies the original goal of combining the best attributes of the k weighting procedures and the multi-dimensional calculations for an optimum long-range fuel management tool.

4.4 Future Work

The development of this method allows for an extensive amount of future work possibilities. The program can be used for an endless number of studies and significant improvements can be made to the program to make it more usable. Some of the basic schemes for adding to the program are fuel cycle cost calculations, burnable poison search procedures and generation of data to verify the loading pattern-cycle length compatability with design considerations.

Presently the program prepares a data set for use in fuel cycle cost calculations. This could be expanded to actually include a fuel cycle cost calculation like MITCOST built into the program. This would greatly simplify the comparison of various proposed fuel management schemes since it would give a direct comparison of fuel cycle costs. This addition could be made relatively easily since the data set is presently prepared. It would require the input of the fuel cost parameters and the coupling to the fuel cost program.

The feasibility of adding options concerning the burnable poisons is intriguing. Since burnable poisons are handled on a region-wise basis it is possible to determine the affect of the region-wise distribution of the burnable poisons in the core. Therefore, it is conceivable that a search procedure could be built into the program to search for the burnable poison distribution which produces the optimum power distribution. Of course, this raises difficulties as to what time in the cycle one wishes to optimize the power distribution and also how to effectively converge on a desired shape. A Haling method may produce good results for the power distribution optimization; that is to compute iteratively a BOC power distribution with

BPs that have the same power distribution as the EOC with depleted BPs.

Another search feature with burnable poisons is the possibility of searching for the correct number to give an acceptable moderator temperature coefficient at the beginning of the cycle. This requires the computation of the BOC critical boron concentration and using some procedure to generate the appropriate moderator temperature coefficient. This scheme is straightforward and should cause no convergence problems. However, the optimum distribution of these burnable poisons may have to be addressed. Thus, these burnable poison search features may have to all be added at once.

The design considerations offer a third different area for program expansion. This area is multifold, since there are many varied design considerations to evaluate. They all require some sort of fits or functions to extrapolate the program data into the controlling variables. These fits may be used to give various coefficients or control rod worths or correction terms to the region power sharings may be possible to generate peak rod powers and burnups. Some of the variables that could possibly be checked are:

moderator temperature coefficients

soluble boron coefficients

power coefficients and defects

total control rod worth

maximum $F_{\Delta H}$ for a rod

maximum assembly burnup

maximum pellet burnup

delayed neutron fraction

and all other various quantities.

It is conceivable that the majority of the design constraints can be roughly estimated for many future cycles. This would allow one to see if the fuel management of one cycle has any long range problems.

The program in its present form can be used for a number of studies. One possibility is the usefulness for BWR fuel management. The program was designed to be able to calculate BWR or PWR cores. However, all the qualification effort was for PWRs. Therefore, methods for using FLAC for BWR fuel management have yet to be formulated and qualified.

The structure of the program allows for feasibility studies to be performed in areas where only advanced and expensive tools could previously be used. For example, cycling studies using many burnable poisons such as 18 month cycles or the in-out-in fuel management strategy. FLAC has the capability of investigating the effects on power distribution and cycle lifetime capability due to the number and placement of burnable poisons. Also, the effects of different burnable poison compositions (i.e. different w/o of B_4C in AlO_3) could be relatively easily studied. Here one has the trade off of rods depleting quickly or slowly and its affect on residual worth and cycle power sharings.

The program can also be used to determine the equilibrium affect of a particular fuel management scheme very easily. The sophisticated methods require numerous cycles in succession to reach a pseudo equilibrium cycle. FLAC has the equilibrium cycle calculation as an option. Thus one can easily determine the equilibrium effect of different types of loading patterns or of splitting the feed region up into various batches of different enrichments. The possibilities are endless, but most could be handled easily by FLAC.

CHAPTER 5

REFERENCES

1. Antonopoulos, P. T. and Turnage, J. C., "Management Analysis of Nuclear Allocation for the Generation of Electricity", Nuclear Technology, Vol.34, p. 347 (August, 1977)

2. Barry, R. F., "The Revised LEOPARD-4 Code - Spectrum Dependent Non-Spatial Depletion Program", WCAP-2579, Westinghouse Electric Corporation (March, 1965)

3. Cadwell, et.al., "PDQ-7 Reference Manual", WAPD-TM-678, Westinghouse Electric Corporation (January, 1967)

4. Delp, D. L. et.al., "FLARE - A Three-Dimensional Boiling Water Reactor Simulator", GEAP-4598, General Electric Company (1964)

5. Forbes, I. A., et.al., "A Rapid Empirical Method for Multi-cycle Fuel Cost Analysis", Trans. Am. Nucl. Soc., Vol. 21, p. 257 (1975)

6. Haling, R. K., "Opèrating Strategy for Maintaining an Optimum Power Distribution Throughout Life", TID-7672, USAEC, (1964)

7. Lang, F. D. and Isakson, R. A., "Computer Application of Cyclic Reactivity Carryover Effects in Fuel Management", Trans. Am. Nucl. Soc., Vol. 13, p. 39 (1970)

8. Naft, B. N., "The Effect of Regionwise Power Sharing on PWR Incore Fuel Management", Trans. Am. Nucl. Soc., Vol. 21, p. 655 (1972)

9. Pilat, E. E., personal communication.

10. Rieck, T. A., "The Effect of Refueling Decisions and Engineering Constraints on the Fuel Management for a Pressurized Water Reactor", Ph.D. Thesis, MIT Department of Nuclear Engineering, February, 1974

11. VerPlanck, D. M. and Ferguson, D. R., "SIMULATE, Reactor Simulator Code, User's Manual", Yankee Atomic Electric Company, (1972); revision (September, 1975)

12. Vogt, D. K. and Carlson, R. W., "A Model for Rapid Evaluation of Fuel Management Options", Trans. Am. Nucl. Soc., Vol. 22, p. 319 (1975)

APPENDIX A

FLAC USER'S MANUAL

FLAC was written to be an easy to use program that could be readily adopted by a utility engineer, vendor engineer or graduate student to his own specific needs. However, to allow flexibility in its use, many options had to be provided. This tends to make the input look quite expansive; where it is very trivial for most cases. Many cases are as small as five cards, while it is possible to generate some relatively large input decks.

Similarly, the output was kept to a minimum to keep the engineer from becoming lost in a maze of numbers. The relavent data is presented in a neat, well labeled format with only digits with some significance editted. It seems rather unnecessary to give the k_{eff} to the nearest tenth of a pcm when it is probably only accurate to the nearest 200 pcm. However, to allow meaningful perturbation calculations, the k_{∞} is given to the nearest 10 pcm.

A.1 Input Preparation

A.1.1 Input Description

The input to FLAC is relatively straight forward and simple. All the input is in fixed format, so particular attention must be paid to positioning of the data on the card. Cases are easily stacked and much of the information can propagate from case to case.

The input can be divided into two distinct parts - the general case data and the cycle by cycle data. The input parameters are all listed in Table A.1 with their corresponding descriptions. This section will

TABLE A-1

INPUT PARAMETERS

COLS.	PARAM.	TYPE	SUBPT	UNITS	DESCRIPTION
****	TITLE CARD (20A4)				
1-80	TITLE	A	(20)	--	FULL CARD OF TITLE DATA FOR CASE
****	CASE DATA CARD A (24I3)				
1- 3	NASS	I	--	--	NUMBER OF ASSEMBLIES IN WHOLE CORE
4- 6	NADIAM	I	--	--	NUMBER OF ASSEMBLIES ACROSS CORE
7- 9	NBU	I	--	--	NUMBER OF BURNUPS IN INTERPOLATION TABLE. IF NOT ENTERED, USE DEFAULT OR PREVIOUS CASE BURNUPS. IF =-1, THEN TABLES INPUT AS COEFFICIENTS OF DEFINED FUNCTIONS AS IN SIMULATE.
10-12	NRICH	I	--	--	NUMBER OF K INF. SETS TO BE READ IN. IF =0, USE DEFAULT OR PREVIOUS CASE DATA, IF >0, REPLACE PREVIOUS SET WITH NEW SET. IF <0, ADD THIS MANY NEW SETS TO PREVIOUS SETS.
13-15	ISO	I	--	--	IF =1, READ IN ISOTOPIC DATA ALONG WITH K INF. DATA. IF =-1, GENERATE ISOTOPIC DATA FROM MUDDLE FIT.

16-18	IFCOST	I	--	--	IF =1, PREPARE DATA SET FOR A FUEL CYCLE COST CALCULATION.
19-21	INFIT	I	--	--	IF =1, READ IN FIT DATA.
22-24	IHAL	I	--	--	IF =1, A HALING CALCULATION WILL BE DONE FOR EACH CYCLE.
25-27	DEBUG	L	--	--	IF =1, DEBUG PRINT OPTION
28-30	NW	I	--	--	NUMBER OF DIFFERENT VALUES OF W TO USE. IF NOT ENTERED, SET TO 1.
31-33	NA	I	--	--	NUMBER OF DIFFERENT VALUES OF ALPHA TO USE. IF NOT ENTERED, SET TO 1.

**** CASE DATA CARD B (12F6.2)

1- 6	W	R	--	--	PROBABILITY THAT A NEUTRON BORN IN AN ASSEMBLY WILL BE ABSORBED IN AN ADJACENT ASSEMBLY. DEFAULT = 0.115.
7-12	ALPHA	R	--	--	ALBEDO ON EXTERIOR EDGE OF CORE. DEFAULT = 0.30.
13-18	ELIM	R	(1)	W/O	MINIMUM FEED ENRICHMENT IN SEARCH
19-24	ELIM	R	(2)	W/O	MAXIMUM FEED ENRICHMENT IN SEARCH IF NOT INPUT, SET TO 4.5
25-30	BUDIF	R	--	GWD/MTM	BURNUP DIFFERENCE FOR SPLIT REGION BURNUP OF SPLIT REGION IS GIVEN BY BU = AVG.BU - BUDIF*(1-N KEPT)/N)
31-36	CYBIAS	R	--	--	K EFF BIAS FACTOR - DIVIDES INTO K
37-42	HEIGHT	R	--	FEET	CORE HEIGHT FOR GENERATION OF AXIAL BIAS FACTOR AS A FCN OF BURNUP.
43-48	ULOAD	R	--	MTM	TOTAL CORE AVERAGE LOADING.
49-54	POWER	R	--	MW T	TOTAL CORE POWER.
55-60	DELW	R	--	--	INCREMENTAL CHANGE IN W TO BE USED TO GENERATE NW-1 VALUES OF W AFTER THE BASE INPUT VALUE.
61-66	DELA	R	--	--	INCREMENTAL CHANGE IN ALPHA TO BE USED TO GENERATE NA-1 VALUES OF ALPHA AFTER THE BASE INPUT VALUE.

**** BURNUP DATA CARDS (12F6.2) (ONLY IF NBU>0)

BURNUP	R	(I)	GWD/MTM	BURNUP ARRAY IN ORDER OF INCREASING BURNUP. 12 ITEMS PER CARD, UP TO 20 ENTRIES READ IN. ONLY READ IN IF NBU>0, THEN NBU ENTRIES READ

**** AXIAL BIAS CARDS (6E12.5) (ONLY IF NBU>0, HEIGHT=0)

BIAS	R	(I)	--	AXIAL BIAS FACTOR AS A FUNCTION BU, ENTERED ONLY IF NBU>0 AND HEIGHT=0. NBU ENTRIES CORRESPONDING TO BURNUP(I), 6 ENTRIES PER CARD.

**** BP WORTH CARDS (6E12.5) (ONLY IF NBU>0)

BP	R	(I)	PCM	WORTH OF ONE BP ROD/ ASSEMBLY AS A FUNCTION OF BURNUP(I). ENTERED ONLY IF NBU>0. NBU ENTRIES, 6 PER CARD.

**** BP COEFFICIENT CARD (6E12.5) (ONLY IF NBU<0)

B	R	(I)	--	COEFFICIENTS OF FUNCTION FOR WORTH OF ONE BP ROD/ASSEMBLY. FUNCTION IS B(1)*(1.0 + B(2)*BU + B(3)*BU**2 + B(4)*BU**3 + B(5)*BU**5) WHERE BU IS THE BURNUP IN GWD/MTU.

```
****  ENRICHMENT CARDS  (12F6.2)          (ONLY IF NRICH NE 0)

      ENRICH   R    (I)    W/O   ENRICHMENT CORRESPONDING TO EACH
                                 DATA SET. W/O U-235 OR W/O PU.
                                 NRICH VALUES ENTERED, 12 PER CARD.

****  FUEL TYPE CARDS   (12I6)           (ONLY IF NRICH NE 0)

      ITYPE    I    (I)    --    FUEL TYPE CORRESPONDING TO EACH
                                 DATA SET.  NRICH VALUES ENTERED 12
                                 PER CARD.

****  POWER/FISSION RATIO CARDS  (12F6.0) (ONLY IF NRICH NE 0)

      POWFIS   R    (I)    MEV/N POWER TO FISSION SOURCE RATIO FOR
                                 EACH FUEL TYPE. (KAPPA/NU) NRICH
                                 VALUES, 12 PER CARD.
```

**** K INF CARDS (6E12.5) (ONLY IF NRICH NE 0, NBU GE 0)

KINF R (I,J) -- K INF CORRESPONDING TO BURNUP (I) FOR ENRICHMENT J. NBU VALUES FOR EACH SET, 6 VALUES PER CARD, NRICH SETS.

**** KINF COEFFICIENT CARDS (6E12.5) (ONLY IF NRICH NE 0, NBU<0)

B R (I) -- COEFFICIENTS OF FUNCTION FOR K INF. AS A FUNCTION OF BURNUP. ONE SET ENTERED FOR EACH ENRICHMENT.
THE FUNCTION IS OF THE FORM:
K INF = B(1)*Z1*Z2
Z1=1.0-B(2) - B(3)*(1.0+B(4)*BU)
Z2=1.0 - B(5)*BU + B(6)*BU**2 +
  B(7)*BU**3 + B(8)*BU**4
WHERE BU IS BURNUP IN GWD/MTU.
NOTE THAT B(1) TO B(8) ARE TO BE ENTERED, SO 2 CARDS/SET ARE NEEDED.

```
****  ISOTOPIC DATA CARDS  (6E12.5)   (ONLY IF ISO=1, NRICH NE 0)

      ISOTOP  R  (I,J,1) KG/MTM URANIUM CONTENT AS A FUNCTION OF
                                BURNUP(I).  NBU ENTRIES, 6 ENTRIES
                                PER CARD FOR ENRICH(J).

      ISOTOP  R  (I,J,2) KG/MTM U-235 CONTENT AS A FUNCTION OF
                                BURNUP(I).  NBU ENTRIES, 6 ENTRIES
                                PER CARD FOR ENRICH(J).

      ISOTOP  R  (I,J,3) KG/MTM TOTAL PU CONTENT AS A FUNCTION OF
                                BURNUP(I).  NBU ENTRIES, 6 ENTRIES
                                PER CARD FOR ENRICH(J).

      ISOTOP  R  (I,J,4)   --   FISSILE/(TOTAL PU) RATIO AS A
                                FUNCTION OF BURNUP(I) FOR ENRICH(J).

         NOTE ***  THE ORDER OF CARDS ARE ALL BURNUPS IN SEQUENCE
                  ON CARD(S) TO FORM A SET.  THEN EACH SET FOR
                  DIFFERENT ISOTOPIC INFORMATION (4 SETS) FOR
                  ENRICH(J) TO FORM A LARGER SET WHICH IS THEN REPEATED
                  FOR EACH ENRICHMENT. IF NOT ENTERED, DEFAULT
                  DATA WILL BE USED FROM FIT.  DATA IS ONLY USED
                  FOR COST DATA SET PREPARATION.
```

**** COEFFICIENT FIT DATA SET (5F6.2/ (6E12.5)/)
(ONLY IF INFIT =1)

1- 6	XEWRTH	R	--	PCM	TOTAL HFP XENON WORTH AT BOL
7-12	DPWRTH	R	--	PCM	TOTAL DOPPLER DEFECT HFP TO HZP AT BOL.
13-18	BWRTH	R	--	PCM/PPM	BORON WORTH AT BOL.
19-24	TEMP	R	(1)	DEG F	HZP TEMPERATURE.
25-30	TEMP	R	(2)	DEG F	AVERAGE CORE HFP TEMPERATURE.
	COFIT	R	(1)	--	MODERATOR TEMPERATURE FIT PARAMETERS UP TO 10 ALLOWED, 6 PER CARD.

**** CYCLE DATA **** ALL OF THE REMAINING DATA IS INPUT FOR EACH CYCLE, THEN FOR THE NEXT CYCLE, ETC. THE CYCLES SHOULD BE IN INCREASING ORDER. ONE MUST START WITH CYCLE 1, BUT BOC BURNUPS MAY BE ENTERED. NO CYCLE MAY BE ELIMINATED FROM THE SEQUENCE, EXCEPT AT THE END. THUS ANY NUMBER OF CYCLES UP TO 20 MAY BE RUN AND THEY MUST BE CONSECUTIVE. CYCLE 21 IS AN EQUILIBRIUM CYCLE AND IS SEPARATE FROM THE REST. THE CYCLE DATA MUST BE FOLLOWED BY A BLANK CARD AFTER THE LAST CYCLE INPUT.

**** CYCLE OPTION CARD (8I3,6F6.0)

1- 3	IS	I	--	--	CYCLE NUMBER 1-20 OR 21 FOR EQUIL. CYCLE.
4- 6	ISURCH	I	(IS)	--	SEARCH PARAMETER FOR CYCLE IS. =1 SEARCH ON CYCLE BURNUP =2 SEARCH ON NUMBER OF BPS =3 DETERMINE EOL K EFF. =10+J SEARCH ON ENRICHMENT FOR FEED J IN CYCLE IS
7- 9	ISHUF	I	(IS)	--	SHUFFLE GENERATION OPTION =0 INPUT SHUFFLE FOR THIS CYCLE =-1 USE SHUFFLE FROM LAST CASE FOR THIS CYCLE. >0 GENERATE SHUFFLE DATA AUTOMATICALLY =1 ASSUME PERITHERAL FEED AND CENTER ASSEMBLY FROM PREVIOUS CYCLE. =2 ASSUME PERIPHERAL FEED AND CENTER ASSEMBLY FROM CYCLE 1.

10-12	NREG	I	(IS)	--	NUMBER OF REGIONS IN CYCLE (<16)
13-15	NFEED	I	(IS)	--	NUMBER OF FEED REGIONS IN CYCLE. MAY BE 0,1,2, OR 3.
16-18	IBOC	I	--	--	=1 IF BOC REGION BURNUPS ARE TO BE ENTERED FOR THIS CYCLE. =0 IF BOC REGION BURNUPS ARE TO BE CALCULATED USING SHUFFLE DATA AND EOC REGION BURNUPS FROM PREVIOUS CYCLES. =-1 IF PREVIOUS CASE BOC REGION BURNUPS ARE TO BE USED THIS CYCLE.
19-21	IFEDGE	I	(IS)	--	=0 INPUT THE EDGE DATA. =-1 USE THE EDGE DATA FROM THE PREVIOUS CASE FOR THIS CYCLE. >0 GENERATE EDGE DATA AUTOMATICALLY =1 ASSUME A RING GEOMETRY IN THE CORE FOR THE EDGE DATA. =2 ASSUME A CHECKERBOARD GEOMETRY FOR THE INTERIOR OF THE CORE.
22-24	DUMMY	I			
25-30	CYLU	R	(IS)	GWD/MTM	DESIRED (OR GUESSED) CYCLE BURNUP.
31-36	BP	R	(IS)	--	NUMBER OF BP RODS (OR GUESS) IN CORE.
37-42	EOLK	R	(IS)	--	DESIRED END OF LIFE K EFF FOR CYCLE. CAN BE VARIED TO CHANGE FOR EOL BORON CONCENTRATION, COASTDOWN OR EARLY SHUTDOWN.

43-48	CAPFAC	R	(IS)	PERCENT	CAPACITY FACTOR FOR CYCLE, EXCLUDING REFUELING SHUTDOWN.
49-54	CYTIM	R	(IS)	DAYS	CYCLE LENGTH, EXCLUDING SHUTDOWN. ONLY 2 OF 3 (CYBU, CAPFAC, CYTIM) NEED BE ENTERED. IF 3 ENTERED, CYTIM IS RECALCULATED.
55-60	SDLEN	R	(IS)	DAYS	REFUELING SHUTDOWN LENGTH. IF NOT ENTERED, SET TO 42.

**** FEED DATA CARDS (2I3,A3,I3,2F6.0)

INPUT FOR EACH REGION IN CYCLE 1 OR JUST THE FEED REGIONS FOR OTHER CYCLES. START WITH THE LOWEST ENRICHMENT REGION AND PUT IN ORDER OF INCREASING ENRICHMENT.

1- 3	NAFEED NAC1	I I	(IS,I) (I)	-- 	NUMBER OF FEED I (OR REGION I IN CYCLE 1) ASSEMBLIES.
4- 6	FTYPE C1TYPE	I I	(IS,I) (I)	-- --	FUEL TYPE FOR FEED I (OR REGION I IN CYCLE 1) CORRESPONDING TO INPUT TABLE OF K INF. INTERPOLATION WILL ONLY BE DONE WITH IDENTICAL FUEL TYPES.
7- 9	FID C1ID	A A	(IS,I) (I)	-- --	ID FOR FEED REGION 2. (IE. 5A,6PU). THIS IS USED FOR LABELING OF OUTPUT.
10-12	DUMMY	I			
13-18	FRICH C1RICH	R R	(IS,I) (I)	W/O W/O	ENRICHMENT OF FEED I IN CYCLE IS OR ENRICHMENT OF REGION I IN CYCLE 1.
19-24	FLOAD C1LOAD	R R	(IS,I) (I)	MTM MTM	LOADING PER ASSEMBLY FOR FEED I IN CYCLE IS OR REGION I IN CYCLE 1. IF NOT ENTERED, SET TO ULOAD/NASS.

**** REGION SIZE CARD (12I6) (ONLY IF ISHUF(IS)=0)

NAREG I (I,IS) -- NUMBER OF ASSEMBLIES IN EACH REGION STARTING WITH THE HIGHEST BURNUP (INNERMOST) REGION. NREG(IS) ENTRIES, 12 PER CARD.

**** SHUFFLE CARD (12I6) (ONLY IF ISHUF(IS)=0)

ISHUF I (I,IS) -- SHUFFLE INICIES SPECIFYING PREVIOUS LOCATION OF EACH REGION(I) IN CYCLE IS. VALUE OF FORM 100*K+L WHERE K IS THE PREVIOUS CYCLE NUMBER AND L IS THE REGION NUMBER IN THE CYCLE. FOR A FEED REGION, K=0 AND L IS THE FEED NUMBER (1,2, OR 3) NREG(IS) ENTRIES, 12 PER CARD.

**** EDGE CARDS (11I6) (ONLY IF IFEDGE(IS)=0)

1- 6	JR	I	--	--	REGION NUMBER CORRESPONDING TO DATA ON THE WHOLE CARD.
7-12	NEDGE	I	(1,K)	--	SPECIFICATIONS OF THE NUMBER OF EDGES
13-18	NEDGE	I	(2,5)	--	ADJOINING REGION JR IN CYCLE IS.
19-24	NEDGE	I	(1,K+1)	--	NEDGE(1,.) IS THE REGION NUMBER
25-30	NEDGE	I	(2,K+1)	--	ADJOINING REGION JR AND NEDGE(2,.)
	ETC.				IS THE NUMBER OF EDGES BETWEEN THEM. FOR EDGES ON THE EXTERIOR, SET NEDGE(1,.)=0. NOTE THAT K IS JUST A COUNTER. ALSO ONLY NON-ZERO NUMBER OF EDGES MUST BE ENTERED AND ONLY ONE SET FOR EACH REGION PAIR. 5 PAIRS MAY BE ENTERED ON EACH CARD.

NOTE: LAST CARD WITH EDGE DATA SHOULD BE FOLLOWED BY A BLANK CARD.

```
****   BP FRACTION CARD   (12F6.2)           (ONLY IF BP(IS) NE 0)

       BPFRAC  R    (I,IS)   --    FRACTION OF BP RODS IN EACH REGION
                                   I IN CYCLE IS.  NREG(IS) ENTRIES,
                                   12 ENTRIES PER CARD.

****   BOC BURNUP CARD    (12F6.2)           (ONLY IF IBOC=1)

       BOCBU   R    (I,IS)GWD/MTM BOC REGION BURNUPS FOR CYCLE IS.
                                 NREG(IS) ENTRIES, 12 ENTRIES / CARD.

****   END OF CYCLE (IS) DATA - REPEAT SET FOR NEXT CYCLE.
****   AFTER LAST CYCLE DATA SET, PUT A BLANK CARD.
****   CASES MAY BE STACKED DIRECTLY.
```

elaborate on those items in Table A.1 which need further clarification or which are very important.

In the table, the variable name as used in the program is given along with the columns where it should be input. The type of the variable is also given (I for integer, R for real and A for alphanumeric). Note that no decimal points may be used with integer data, but that they should be used with real data. The subscripts, if any are given and the units appropriate to the code are also specified along with a description of the input quantity.

Sample cases are given in Section A.1.2 for an example. The cases are a base case with sequential cycles followed by two equilibrium cycle cases. These cases demonstrate many of the input options such as shuffle data, edge data and BOC region burnup input. The cases are marked up to identify the various parameters.

Every case begins with a title card. This card is used to title the output and as such appears at the top of each page.

The title card is followed by case data cards A and B. Card A contains integer data while card B contains real data. Only the first two parameters on cards A and B are required. The parameters NBU, NRICH, ISO and INFIT specify if specific sets of input are also to be entered. NBU and NRICH must be entered for the first case in a set of cases since the k_∞ table must be input in some manner. The debug print option should be used only when necessary to determine the cause of an undetermined problem. This option prints a significant amount of intermediate data out chiefly through the subroutine BUGOUT which is a generalized routine for printing arrays.

The values of W and α on card B are the two most important quantities to be entered. W is the W_j parameter in the basic FLARE equation and α is the albedo. W by definition is the probability that a neutron born in an assembly will be absorbed in a single adjacent assembly. W usually has a value in the range of 0.10 to 0.14 and α usually has a value in the range of 0.25 to 0.40. See Section 3.3.1 for a discussion on the sensitivity of these parameters.

If a sensitivity study on W and α is desired, the variables NW, NA, DELW and DELA should be entered on case data cards A and B. This will produce a NWxNA matrix of cases using different combinations of W and α. The first case uses the input values of W and α and succeeding cases change the relative values by DELW and DELA. Thus one can easily perform a sensitivity study on the combinations of W and α.

Also, on case data card B are the minimum and maximum enrichments allowed in an enrichment search. From experience, limits on the feed enrichment during a search are necessary to preclude wild fluctuations which could possibly occur in the feed enrichment. The variable BUDIF allows one to have a uniform burnup distribution over a specified burnup range to enable split regions to discharge a higher burnup than the region average. This option is discussed fully in Section 2.8. The cycle bias factor is an adjustment term which is applied to each cycle. Each k_∞ is divided by this cycle bias factor. The core height is used to calculate the axial leakage effect using a default curve for 12 foot cores. The loading and power are used to convert from burnup units to time units and do not have to be entered unless appropriate cycle times are desired.

The burnup array is entered only on option. All tabular data uses this array as a base and therefore, all the data must be consistent with it. The default set is the data 0, 4, 8, ..., 48 GWD/MTU.

The BIAS input is generally used only to account for the effects of axial leakage effects as a function of burnup. However, it can be used as a general bias factor depending on the need. The region k_∞ are divided by this bias factor.

The burnable poison worth can be entered in a functional form or in a tabular form. In either case the data is for the worth of one BP rod in a representative assembly as a function of burnup. This should include the worth of the poison, the displacement of moderator and the non-poison contribution such as clading or filler.

The input for the different fuel types is input next. Note that the data can add to existing data or create a new data set and also that the data can be input in tabular or functional form. The enrichment and fuel type are used to denote each different fuel data set. The enrichment can be used to denote the w/o of U-235, the w/o of plutonium or the w/o of U-233 or any other parameter (such as H/u ratio) which may want to be varied. It is best to maintain some deviation between enrichments when interpolation will be done. At least 0.1 w/o difference between enrichments of the same fuel type is recommended to insure that the secondary parameters do not become important and distort the true variation of k_∞ as a function of enrichment. The fuel type allows one to differentiate between different fuel data sets where another parameter other than enrichment is important. This could correspond to different vendor's fuel, stainless versus zircaloy clad or different recycle batches of plutonium fuel. The power to fission ratio is important only for significantly different fuel types such as uranium and plutonium fuel. This corresponds to an average value of κ/ν, but since it is only a weighting function, any multiple will suffice. For example, with all uranium fuel, an input of 1.00 for all enrichments is fine.

The k_∞ data may be entered as coefficients in a polynomial function or in a tabular form. In any case the k_∞ data should include xenon, samarium and full power doppler, but they should not contain any soluble boron.

The isotopic information can either be ignored, default generated or it can be input. The only use is to be interpolated at the discharge burnup for each batch for input to a fuel cost program. The data as input is interpolated, so the information should be in the desired end units.

The coefficient data is presently not used. The only input which has any effect is the boron worth entry which serves as a flag. When the boron worth is specified, each depletion step will include a search for a uniformly distributed poison to maintain core criticality.

Separate cycle data must be entered for each cycle for which a calculation is to be performed. Cycle 21 is an equilibrium cycle calculation and is independent of the other cycles. One can do just an equilibrium cycle, just a sequential series of cycles or both. The sequential series must always start with input for cycle 1 and include all cycles until the last one desired. The beginning of cycles region burnups may be entered for any cycle including the first. So cycle 1 may in fact be a later cycle with specified beginning of cycle region burnups. For cycle 1, no shuffle specification is needed, but the information for each region must be input. For succeeding cycles, only the information for the feed regions need to be entered. This information consists of the number of assemblies, an identifying ID, the fuel type and enrichment and the assembly average loading if it is different from the reference core average.

The search parameter can vary between cycles. The search for the cycle burnup allows one to determine what burnup capability exists for a particular cycle. The search for the end of cycle k_{eff} is generally used for verification when the cycle burnup and feed enrichments are well known. The feed enrichment search allows the program to calculate the feed enrichment to produce the required cycle lifetime. This search procedure should be used with care since one is trying to influence the whole core with a change in only a fraction of the core. Large fluctuations in enrichment are possible.

The EOC k_{eff} which is desired is an input variable. This allows one to account for an early shutdown or a coastdown by inputting values respectively greater than 1.000 and less than 1.000.

The cycle burnup, cycle time and capacity factor are three variables; only two of which are independent. If any two are entered, the third is calculated. If all three are entered, the cycle time will be recalculated. Note that the cycle time and capacity factor do not include the refueling shutdown which is handled separately. The only variable which is actually used in the calculation is the cycle burnup. The other parameters are calculated only for editting and fuel cost input preparation.

The fuel shuffle can normally be calculated by the program. The program assumes that the assemblies are shuffled directly from the previous cycle. The feed regions are the highest number regions and the regions from the previous cycle are kept in the same order. The discharged assemblies are taken from region 1 first, then region 2, and so forth until the number of discharged assemblies in the previous cycle equals the number of feed assemblies. The automated shuffle will not work when assemblies are being reinserted after at least one cycle in the spent fuel pit or when the lowest number region is not to be discharged. Also if the program is to calculate the edge data, it assumes that the regions are ordered in the manner such that the lowest number is at the center and the highest number is furthest out in the periphery. Therefore, if any assemblies other than feed assemblies are desired in the periphery, an explicit shuffle must be input.

When the shuffle is input both the number of assemblies in each region and where the region came from has to be entered. The description of the previous location is given by a number of the form 100 + K + J where K is the cycle number and J is the region number. For example, 903 implies that that region came from region 3 in cycle 9. The code will accept any cycle number less than the current cycle number and any number of assemblies may be transferred as long as it doesn't exceed the number available. For the equilibrium cycle, K must always be 21. The feed regions are also entered but in this case K is zero and J is the feed number (1, 2 or 3). If assemblies are to come from another unit, they should be treated as feed regions with the BOC burnup also input.

The edge data can also be generated automatically by the code. The edge data consists of the number of assembly sides of a region adjoining another specified region. The number of total edges is four times the number of assemblies in a region. Also, the number of edges between two regions is the same independent of the reference region. Some edges also are on the exterior of the core. The total number of these edges is four times the number of assembly rows across the core. The data is given by the reference region, then the adjoining regions and the number of common edges. The periphery is entered as region zero. A blank card is entered to signify the end of the edge data information.

The beginning of cycle burnups may be entered on option. Any non-zero entries will over-ride any calculated burnups. Thus the burnups need only be entered for those regions whose burnups are to be specified. The other regions will use the calculated values. The beginning of cycle burnup are not used in an equilibrium cycle calculation, and hence their input will be ignored.

The cycle data is input on a cycle by cycle basis. A blank card is placed after the last cycle input to signify the end of the data for that case.

Cases may be stacked. The tabular data is saved from case to case and options exist to save some of the other data between cases. All other data must be reentered.

A.1.2 Sample Input

A small input deck has been generated to demonstrate the use of many of the various input options. The simple deck consists of five stacked cases and is listed on the following pages. These cases were also used to generate the sample output described in Section A.2.2.

The first case reads in the tabular data for three enrichments and four burnups. Generally, more burnup steps should be used, but for clarity only four were used in this case. The first case calculates the equilibrium cycle enrichment corresponding to a cycle length of 11.0 GWD/MTM and an end of cycle k_{eff} of 1.0010 which would correspond to residual soluble boron of 10 ppm. The edge data is input in this case.

```
FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - ENRICH SEARCH
157 15   4   3 -1   0   1
 .115   .30                            1.                          .005  .05
 1.      8.     20.     36.
 1.0            1.0           1.0            1.0
 900.           350.          25.           20.
 2.6    2.9     3.1

 1.0    1.0     1.0
 1.24408       1.15416       1.02446       0.88846
 1.26936       1.18096       1.05247       0.91329
 1.28405        1.19698      1.06978       .92983
                -10.


 21  11   1   4   1   0   0   0  11.    0.0    1.0010
 52   0  EQ   0    3.2
      1       3      4
      2       3    180      4     28
      3       4     24
      4       0     60      4     96


FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - 3.10 W/O
157 15
 .115   .30
 21   1   1   4   1   0  -1   0  11.    0.     1.001
 52   0  EQ   0    3.1

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD - AUTO , 3.20 W/O
157 15
 .115   .30
 21   1   1   4   1   0   2   0  11.    0.     1.001
 52   0  EQ   0    3.2
```

```
FLAC TEST CASE - 3 LOOP, 3 REGION RING AUTO - 3.20 W/O
157 15
.115  .30
21   1   1   4   1   0   1   0  11.    0.     1.001
52   0  EQ   0    3.2

3 LOOP CORE - CYCLES 1-8   2.1/2.6/3.1 CY 1, 3.2 FEED - 3 REGION
157 15   0   0   0   0   0   0   0
.115  .30
 1   3   1   3   3   0   2   0   14.5 900.   1.001
53   0   1   0  2.1
52   0   2   0  2.6
52   0   3   0  3.1
 0.0   0.80   0.20
 2   1   1   4   1   0   2   0   11.      0  1.001
52   0   4   0  3.2
 3   1   1   4   1   0   2   0   11.      0  1.001
52   0   5   0  3.2
 4   1   1   4   1   0   2   0   11.      0  1.001
52   0   6   0  3.2
 5   1   1   4   1   0   2   0   11.      0  1.001
52   0   7   0  3.2
 6   1   1   4   1   0   2   0   11.      0  1.001
52   0   8   0  3.2
 7   1   1   4   1   0   2   0   11.      0  1.001
52   0   9   0  3.2
 8   1   1   4   1   0   2   0   11.      0  1.001
52   0  10   0  3.2
```

The second case is also an equilibrium cycle calculation, but the search is on cycle burnup given a 3.10 w/o feed. The edge data from the previous case is used in this case.

The third case is an equilibrium cycle searching on the cycle burnup, but with a feed of 3.20 w/o and the edge data generated automatically assuming a checkerboard interior.

The fourth case is the same as the third case except that a ring interior is assumed for the generation of the edge data.

The fifth case consists of cycles 1 through 8 starting with initial enrichments of 2.1, 2.6 and 3.1 w/o in cycle 1 and feeding 52 assemblies at 3.2 w/o each cycle afterwards. Cycle 1 has burnable poisons and a search on k_{eff} is made. In succeeding cycles, a burnup search is performed. In each cycle, the edge data is generated automatically assuming a checkerboard interior.

A.2 Program Output

A.2.1 Output Description

The output in FLAC is relatively straight-forward and easily understood. There are four different parts to the input - the case data summary, the tabular data edit, the cycle summary and the region summary. A sample output is given in Section A.2.2.

The case data summary merely edits the case input data in a convenient format. The leading lines are printed on the top of each page and give the page number, program name and the case title.

The tabular data is editted next, but it is only given with the first case to reduce redundant printing. The tabular data edits the k_∞ as a function of burnup and enrichment, the BP worth and axial bias factor as a function of burnup and the isotopic data as a function of burnup and enrichment. The data is editted in the same units as entered.

The cycle summary consists of a separate page summarizing each cycle which was calculated in the case. The first items given are the cycle number and which parameter was to be calculated that cycle (EOL k_{eff}, burnup, feed enrichment). Then the general cycle data is presented. This data includes the cycle burnup, the effective full power hours (EFPH), the elapsed cycle time and corresponding capacity factor, the end of life k_{eff}, the number of burnable poisons and the total core loading.

The batch average data is given next. This data presents the data pertinent to each batch. This includes the enrichment, fuel type and number of assemblies in each batch. The number of assemblies discharged from each batch at the end of the cycle is also given. A negative number here implies that this batch is being used in succeeding cycles, but that more assemblies are being used than are available. Thus, in most cases a negative number signifies an error in the input. The source is just the shuffle input describing the cycle and region from which that batch came. The region ID is the region identifier which is input for each feed region. The region average beginning of cycle and end of cycle burnups are given for each batch in the core.

The depletion data gives the core k_{eff} and region power sharings as a function of burnup. The data given corresponds to the indicated burnup and the power sharings are used to deplete the region over the next time step. The power sharings are normalized to a core average value of 1.000.

The edge data summarizes the region to region coupling that was being used for that cycle's analysis: for each region, the total number of edges available (four times the number of assemblies), the number of edges used and the number of edges adjacent to the exterior and each region. A mismatch between the number of edges available and the total number of edges used indicates an input error.

The region summary lists each region and gives data pertinent for a fuel cost calculation. Each region is given along with the cycle in which it was a feed and the initial enrichment and fuel type. The region discharge burnup is calculated by averaging the discharge burnups of each discharge batch corresponding to that region. The individual batches are also given. A batch is here referred to as that part of a region which is discharged at the same time. Each batch is indicated by the cycle and region from which it was discharged. Corresponding to each batch, the burnup and elasped time along with the discharge isotopic information is given. The elapsed time includes the cycle time and refueling shutdown time from when the fuel was initially inserted to when it was discharged and includes any intermediate time in the spent fuel pit. The discharge isotopic information is interpolated from the tabular data in the same manner as the k_∞ given the enrichment, fuel type and discharge burnup.

When the last case has been completed, a message is printed out and the program exits.

A debug option exists which will print out a significant amount of intermediate data. Most of the data is editted using the subroutine BUGOUT for convenience. The BUGOUT edit always includes the array name and subscripts, so determination of the variable corresponding to the debug data is not difficult.

A.2.2 Sample Output

Listed on the following pages are sections of the output corresponding to the sample input. Only representative edits are given, but the case titles are given for future comparison purposes.

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - ENRICH SEARCH

```
NUMBER OF ASSEMBLIES....    157
ASSEMBLIES ACROSS CORE..     15
CORE HEIGHT.............   0.0  FT.
AVERAGE CORE LOADING....   1.00 MTM
CORE THERMAL POWER......     1. MW

PROB. OF ABSORP. (W).... 0.115
ALBEDO.................. 0.300
MINIMUM ENRICHMENT...... 0.0    W/O
MAXIMUM ENRICHMENT...... 4.500  W/O
BURNUP VARIATION PARAM.. 0.0    GWD/MTM
CYCLE BIAS FACTOR....... 1.000

BORON WORTH............. -10.0 PCM/PPM
DOPPLER DEFECT..........    0. PCM
XENON DEFECT............    0. PCM
```

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - ENRICH SEARCH

K EFF VERSUS BURNUP AND ENRICHMENT

ENRICHMENT	2.600	2.900	3.100
FUEL TYPE	0	0	0
POWER/FIS.	1.000	1.000	1.000

BURNUP	BIAS	BP WRTH		K EFF	
1.000	1.0000	900.00	1.24408	1.26936	1.28405
8.000	1.0000	350.00	1.15416	1.18096	1.19698
20.000	1.0000	25.00	1.02446	1.05247	1.06978
36.000	1.0000	20.00	0.88846	0.91329	0.92983

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - ENRICH SEARCH

ISOTOPIC DATA (U) VERSUS BURNUP AND ENRICHMENT

	2.600	2.900	3.100
1.000	0.9984	0.9984	0.9984
8.000	0.9878	0.9878	0.9878
20.000	0.9712	0.9712	0.9712
36.000	0.9524	0.9524	0.9524

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - ENRICH SEARCH

ISOTOPIC DATA (U-35) VERSUS BURNUP AND ENRICHMENT

	2.600	2.900	3.100
1.000	2.4826	2.7819	2.9814
8.000	1.7934	2.0757	2.2653
20.000	1.0048	1.2292	1.3838
36.000	0.4222	0.5558	0.6520

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - ENRICH SEARCH

ISOTOPIC DATA (PU) VERSUS BURNUP AND ENRICHMENT

	2.600	2.900	3.100
1.000	0.5424	0.5320	0.5257
8.000	3.3313	3.3194	3.3105
20.000	5.6175	5.7103	5.7629
36.000	6.6372	6.8515	6.9805

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - ENRICH SEARCH

CYCLE 21

SEARCH PARAMETER..FEED 1

BURNUP............ 11.000 GWD/MTM
EFPH..............264000. HOURS
CYCLE TIME........13750.0 DAYS
SHUTDOWN LENGTH... 42.0 DAYS
CAPACITY FACTOR... 80.0 PERCENT
EOL KEFF.......... 1.0011
NUMBER OF BPS..... 0.
LOADING........... 1.000 MTM

BATCH DATA

	1	2	3	4
ENRICHMENT	3.211	3.211	3.211	3.211
FUEL TYPE	0	0	0	0
ASSEMBLIES	1	52	52	52
DISCHARGED	1	51	0	0
SOURCE-C,R	2102	2103	2104	1
REGION ID	EQ	EQ	EQ	EQ
BOC BURNUP	33.036	22.320	10.093	0.0
EOC BURNUP	42.159	33.036	22.320	10.093

DEPLETION DATA

BOS BURNUP	K EFF	1	2	3	4
0.0	1.1137	0.801	0.961	1.110	0.933
4.000	1.0707	0.833	0.976	1.112	0.916
8.000	1.0299	0.862	0.988	1.112	0.903
11.000	1.0011	0.878	0.994	1.108	0.900

EDGE DATA

REGION	AVAIL.	TOTAL	EXTERIOR	1	2	3	4
1	4	4	0	0	0	4	0
2	208	208	0	0	0	180	28
3	208	208	0	4	180	0	24
4	208	208	60	0	28	24	56

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - ENRICH SEARCH

REGION SUMMARY

REGION	FEED IN CYCLE		ENRICH	TYPE	ASMBLY	DIS BU	CYCLE	ASMBLY	DIS BU
EQ	1	21	3.210	0	52	33.212	21	1	42.159
							21	51	33.036

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD CORE - 3.10 W/O

CYCLE 21

SEARCH PARAMETER..BURNUP

```
BURNUP............. 10.554 GWD/MTM
EFPH..............253305. HOURS
CYCLE TIME........13193.0 DAYS
SHUTDOWN LENGTH...   42.0 DAYS
CAPACITY FACTOR...   80.0 PERCENT
EOL KEFF.......... 1.0010
NUMBER OF BPS.....     0.
LOADING...........  1.000 MTM
```

BATCH DATA

	1	2	3	4
ENRICHMENT	3.100	3.100	3.100	3.100
FUEL TYPE	0	0	0	0
ASSEMBLIES	1	52	52	52
DISCHARGED	1	51	0	0
SOURCE-C,R	2102	2103	2104	1
REGION ID	EQ	EQ	EQ	EQ
BOC BURNUP	31.697	21.337	9.541	0.0
EOC BURNUP	40.565	31.697	21.337	9.541

DEPLETION DATA

BOS BURNUP	K EFF	1	2	3	4
0.0	1.1107	0.814	0.971	1.118	0.915
4.000	1.0671	0.845	0.984	1.117	0.902
8.000	1.0259	0.873	0.995	1.116	0.892
10.555	1.0010	0.886	0.999	1.112	0.891

EDGE DATA

REGION	AVAIL.	TOTAL	EXTERIOR	1	2	3	4
1	4	4	0	0	0	4	0
2	208	208	0	0	0	180	28
3	208	208	0	4	180	0	24
4	208	208	60	0	28	24	96

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION CHECKERBOARD - AUTO , 3.20 W/O

CYCLE 21

SEARCH PARAMETER..BURNUP

BURNUP.............	10.934 GWD/MTM
EFPH...............	262409. HOURS
CYCLE TIME.........	13667.1 DAYS
SHUTDOWN LENGTH...	42.0 DAYS
CAPACITY FACTOR...	80.0 PERCENT
EOL KEFF...........	1.0010
NUMBER OF BPS.....	0.
LOADING...........	1.000 MTM

BATCH DATA

	1	2	3	4
ENRICHMENT	3.200	3.200	3.200	3.200
FUEL TYPE	0	0	0	0
ASSEMBLIES	1	52	52	52
DISCHARGED	1	51	0	0
SOURCE-C,R	2102	2103	2104	1
REGION ID	EQ	EQ	EQ	EQ
BOC BURNUP	32.839	22.238	10.112	0.0
EOC BURNUP	41.909	32.839	22.238	10.112

DEPLETION DATA

BOS BURNUP	K EFF	1	2	3	4
0.0	1.1131	0.801	0.955	1.107	0.941
4.000	1.0700	0.834	0.972	1.110	0.922
8.000	1.0292	0.862	0.986	1.110	0.907
10.934	1.0010	0.879	0.992	1.107	0.904

EDGE DATA

REGION	AVAIL.	TOTAL	EXTERIOR	1	2	3	4
1	4	4	0	0	0	4	0
2	208	208	0	0	14	171	23
3	208	208	0	4	171	4	29
4	208	208	60	0	23	29	96

**** FLAC ****

FLAC TEST CASE - 3 LOOP, 3 REGION RING AUTO - 3.20 W/O

CYCLE 21

SEARCH PARAMETER..BURNUP

```
BURNUP............. 10.353 GWD/MTM
EFPH..............248463. HOURS
CYCLE TIME........12940.8 DAYS
SHUTDOWN LENGTH...   42.0 DAYS
CAPACITY FACTOR...   80.0 PERCENT
EOL KEFF.......... 1.0010
NUMBER OF BPS.....     0.
LOADING...........  1.000 MTM
```

BATCH DATA

	1	2	3	4
ENRICHMENT	3.200	3.200	3.200	3.200
FUEL TYPE	0	0	0	0
ASSEMBLIES	1	52	52	52
DISCHARGED	1	51	0	0
SOURCE-C,R	2102	2103	2104	1
REGION ID	EQ	EQ	EQ	EQ
BOC BURNUP	31.142	23.675	10.940	0.0
EOC BURNUP	37.033	31.142	23.675	10.940

DEPLETION DATA

BOS BURNUP	K EFF	1	2	3	4
0.0	1.1164	0.463	0.626	1.262	1.122
4.000	1.0676	0.593	0.745	1.224	1.039
8.000	1.0243	0.708	0.844	1.186	0.975
10.352	1.0010	0.757	0.883	1.169	0.953

EDGE DATA

REGION	AVAIL.	TOTAL	EXTERIOR	1	2	3	4
1	4	4	0	0	4	0	0
2	208	208	0	4	168	36	0
3	208	208	0	0	36	120	52
4	208	208	60	0	0	52	96

**** FLAC ****

3 LOOP CORE - CYCLES 1-8 2.1/2.6/3.1 CY 1, 3.2 FEED - 3 REGION

CYCLE 1

SEARCH PARAMETER..EOL KEFF

BURNUP............	14.500	GWD/MTM
EFPH..............	347999.	HOURS
CYCLE TIME........	18125.0	DAYS
SHUTDOWN LENGTH...	42.0	DAYS
CAPACITY FACTOR...	80.0	PERCENT
EOL KEFF..........	1.0138	
NUMBER OF BPS.....	900.	
LOADING...........	1.000	MTM

BATCH DATA

	1	2	3
ENRICHMENT	2.100	2.600	3.100
FUEL TYPE	0	0	0
BP FRAC.	0.0	0.800	0.200
ASSEMBLIES	53	52	52
DISCHARGED	52	0	0
SOURCE-C,R	1	2	3
REGION ID	1	2	3
BOC BURNUP	0.0	0.0	0.0
EOC BURNUP	16.724	16.052	10.681

DEPLETION DATA

BOS BURNUP	K EFF	1	2	3
0.0	1.1473	1.215	1.072	0.709
4.000	1.1126	1.161	1.108	0.728
8.000	1.0768	1.121	1.128	0.749
12.000	1.0377	1.094	1.129	0.775
14.500	1.0138	1.080	1.128	0.790

EDGE DATA

REGION	AVAIL.	TOTAL	EXTERIOR	1	2	3
1	212	212	0	18	171	23
2	208	208	0	171	8	29
3	208	208	60	23	29	96

**** FLAC ****

3 LOOP CORE - CYCLES 1-8 2.1/2.6/3.1 CY 1, 3.2 FEED - 3 REGION

CYCLE 2

SEARCH PARAMETER..BURNUP

BURNUP............	10.378	GWD/MTM
EFPH..............	249076.	HOURS
CYCLE TIME........	12972.7	DAYS
SHUTDOWN LENGTH...	42.0	DAYS
CAPACITY FACTOR...	80.0	PERCENT
EOL KEFF..........	1.0010	
NUMBER OF BPS.....	0.	
LOADING...........	1.000	MTM

BATCH DATA

	1	2	3	4
ENRICHMENT	2.100	2.600	3.100	3.200
FUEL TYPE	0	0	0	0
ASSEMBLIES	1	52	52	52
DISCHARGED	1	51	0	0
SOURCE-C,R	101	102	103	1
REGION ID	1	2	3	4
BOC BURNUP	16.724	16.052	10.681	0.0
EOC BURNUP	26.007	26.165	22.013	9.711

DEPLETION DATA

BOS BURNUP	K EFF	1	2	3	4
0.0	1.1106	0.878	0.964	1.089	0.949
4.000	1.0663	0.898	0.977	1.093	0.932
8.000	1.0245	0.916	0.987	1.095	0.920
10.378	1.0010	0.922	0.991	1.093	0.918

EDGE DATA

REGION	AVAIL.	TOTAL	EXTERIOR	1	2	3	4
1	4	4	0	0	0	4	0
2	208	208	0	0	14	171	23
3	208	208	0	4	171	4	29
4	208	208	60	0	23	29	96

**** FLAC ****

3 LOOP CORE - CYCLES 1-8 2.1/2.6/3.1 CY 1, 3.2 FEED - 3 REGION

CYCLE 3

SEARCH PARAMETER..BURNUP

BURNUP............	10.823	GWD/MTM
EFPH..............	259763.	HOURS
CYCLE TIME........	13529.3	DAYS
SHUTDOWN LENGTH...	42.0	DAYS
CAPACITY FACTOR...	80.0	PERCENT
EOL KEFF..........	1.0010	
NUMBER OF BPS.....	0.	
LOADING...........	1.000	MTM

BATCH DATA

	1	2	3	4
ENRICHMENT	2.600	3.100	3.200	3.200
FUEL TYPE	0	0	0	0
ASSEMBLIES	1	52	52	52
DISCHARGED	1	51	0	0
SOURCE-C,R	202	203	204	1
REGION ID	2	3	4	5
BOC BURNUP	26.165	22.013	9.711	0.0
EOC BURNUP	35.232	32.417	21.766	10.044

DEPLETION DATA

BOS BURNUP	K EFF	1	2	3	4
0.0	1.1124	0.813	0.947	1.113	0.944
4.000	1.0592	0.842	0.964	1.115	0.925
8.000	1.0283	0.863	0.978	1.114	0.910
10.824	1.0010	0.881	0.984	1.111	0.907

EDGE DATA

REGION	AVAIL.	TOTAL	EXTERIOR	1	2	3	4
1	4	4	0	0	0	4	0
2	208	208	0	0	14	171	23
3	208	208	0	4	171	4	29
4	208	208	60	0	23	29	96

**** FLAC ****

3 LOOP CORE - CYCLES 1-8 2.1/2.6/3.1 CY 1, 3.2 FEED - 3 REGION

CYCLE 4

SEARCH PARAMETER..BURNUP

BURNUP.............	11.107	GWD/MTM
EFPH...............	266572.	HOURS
CYCLE TIME.........	13884.0	DAYS
SHUTDOWN LENGTH...	42.0	DAYS
CAPACITY FACTOR...	80.0	PERCENT
EOL KEFF..........	1.0010	
NUMBER OF BPS.....	0.	
LOADING...........	1.000	MTM

BATCH DATA

	1	2	3	4
ENRICHMENT	3.100	3.200	3.200	3.200
FUEL TYPE	0	0	0	0
ASSEMBLIES	1	52	52	52
DISCHARGED	1	51	0	0
SOURCE-C,R	302	303	304	1
REGION ID	3	4	5	6
EOC BURNUP	32.417	21.766	10.044	0.0
EOC BURNUP	41.512	32.593	22.368	10.209

DEPLETION DATA

BOS BURNUP	K EFF	1	2	3	4
0.0	1.1150	0.790	0.961	1.109	0.934
4.000	1.0719	0.822	0.977	1.110	0.916
8.000	1.0310	0.851	0.990	1.110	0.903
11.107	1.0010	0.869	0.996	1.107	0.900

EDGE DATA

REGION	AVAIL.	TOTAL	EXTERIOR	1	2	3	4
1	4	4	0	0	0	4	0
2	208	208	0	0	14	171	23
3	208	208	0	4	171	4	29
4	208	208	60	0	23	29	96

**** FLAC ****

3 LOOP CORE - CYCLES 1-8 2.1/2.6/3.1 CY 1, 3.2 FEED - 3 REGION

REGION SUMMARY

REGION	FEED	IN CYCLE	ENRICH	TYPE	ASMBLY	DIS BU	CYCLE	ASMBLY	DIS BU
1	1	1	2.100	0	53	16.899	1	52	16.724
							2	1	26.007
2	2	1	2.600	0	52	26.339	2	51	26.165
							3	1	35.232
3	3	1	3.100	0	52	32.592	3	51	32.417
							4	1	41.512
4	1	2	3.200	0	52	32.767	4	51	32.593
							5	1	41.636
5	1	3	3.200	0	52	33.050	5	51	32.876
							6	1	41.954
6	1	4	3.200	0	52	33.024	6	51	32.849
							7	1	41.916
7	1	5	3.200	0	52	32.995	7	51	32.820
							8	1	41.888
8	1	6	3.200	0	52	32.841	8	52	32.841
9	1	7	3.200	0	52	22.232	8	52	22.232
10	1	8	3.200	0	52	10.111	8	52	10.111

A.3 Programmer's Information

A.3.1 Program Routines

The FLAC program is a relatively simple and straight forward computer program. It consists of 14 subroutines and functions arranged in basically a linear fashion. The flow-chart of the subroutines is depicted in Figure A.1.

The program is written exclusively in FORTRAN IV and should be operable on either IBM or CDC machines.

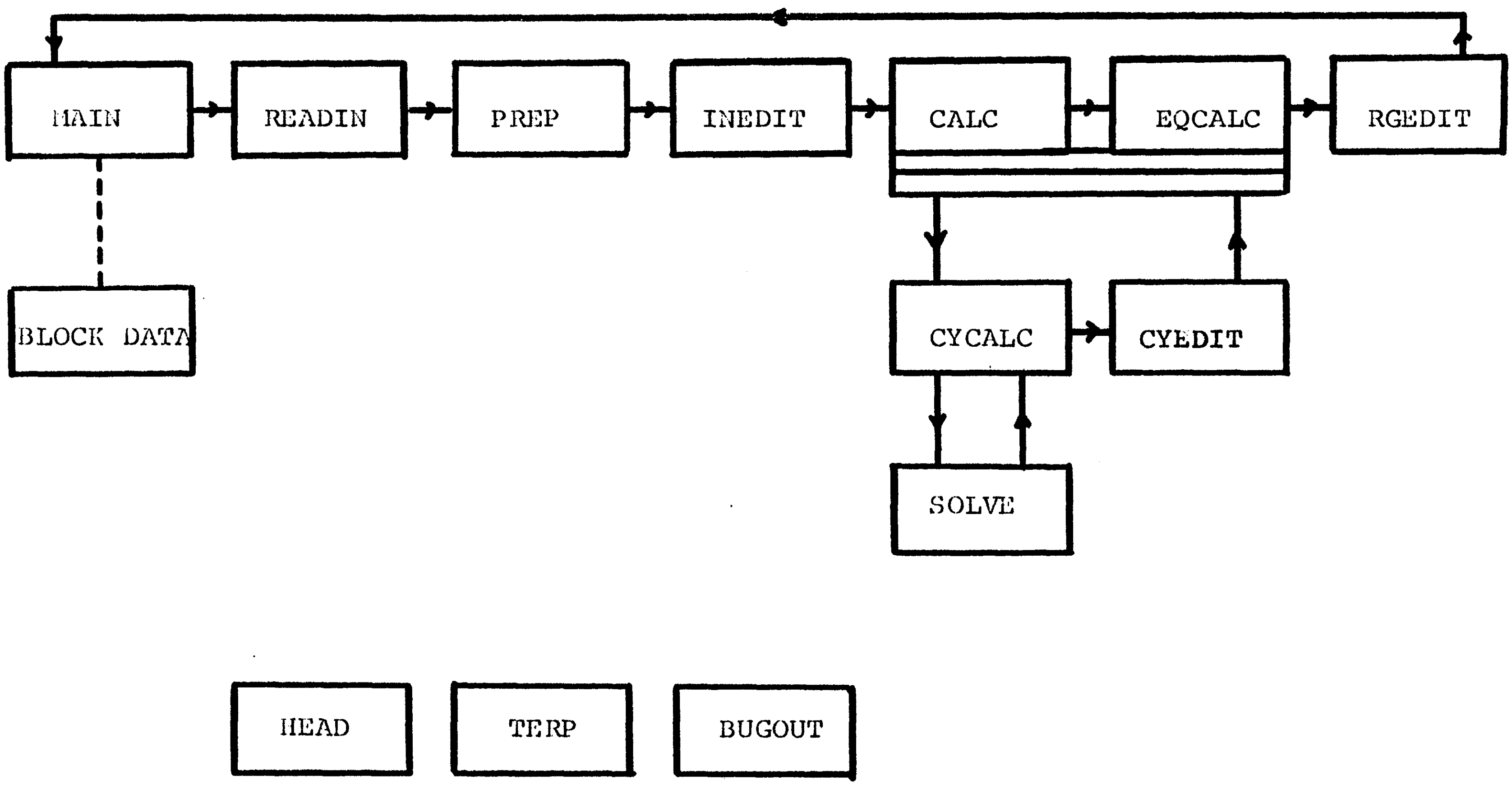
The subroutine MAIN is merely a controller for the program. The block DATA routine supplies the default data. All of the input data is entered through the routine READIN. READIN uses a fixed format input with all the formats listed at the end of the subroutine. Automatic generation of data is accomplished in the subroutine PREP. Here the shuffle array and edge data can be generated. Also in this routine is the calculation of the axial leakage correction factor, the fuel isotopic data fit and the number of discharge assemblies from each region in each cycle. The subroutine INEDIT is then called to edit the basic input information. This completes the problem setup. The next group of subroutines actually solve the problem.

The subroutine CALC sets up the calculation for each cycle except the equilibrium cycle and calls the subroutine CYCALC which does the calculation. For each cycle, CALC sets up the beginning of cycle burnup and the region parameters (enrichment, fuel type, number of assemblies). After the calculation for each cycle has been completed, the subroutine CYEDIT is called to edit the cycle data.

The equilibrium cycle calculation is performed in the subroutine EQCALC. It performs the same function as CALC only for the equilibrium cycle. Also, the search procedure for the desired parameter is carried out

Figure A-1

Program Flowchart



in EQCALC since the parameters for the equilibrium cycle are so interlocking. EQCALC also calls CYCALC and CYEDIT.

The actual cycle calculation is done in CYCALC. Here the matrix is set up, the cycle depletion performed and the search for the desired parameter is accomplished. CYCALC is set up to do a straight depletion or a Haling calculation. It calls the function TERP to supply the k_∞ at the desired enrichment and burnup and calls SOLVE to actually solve the eigenvalue problem.

The subroutine CYEDIT edits the cycle data. The only calculations performed are the calculations of capacity factor or cycle time and the effective full power hours for editting purposes.

After the calculation has been performed for each cycle, the routine RGEDIT is called to generate the region edit. In this routine the fuel cost data set is generated, so coupling with a fuel cost code would be initiated in this subroutine.

Two output routines are called throughout the program. One is HEAD which puts a page heading with the case title at the top of each page. The other routine is BUGOUT which is used extensively for debug printing. BUGOUT only is executed if the debug flag is on. It prints out real or integer variables by specifying the array name, appropriate subscripts and the data.

A.3.2 Program Variables

All of the important variables are in labeled common blocks. The same variable names are used in every subroutine where data is passed through common. All the common blocks and variables in them are listed in Table A.2 along with a description of the variable. Standard FORTRAN typing nomenclature is used except where noted. All the real variables are single precision except for the variables in the subroutine SOLVE which are double precision. The three real variables transferred to SOLVE in the call are also double precision variables.

TABLE A.2

PROGRAM VARIABLES

Variable	Dimension	Description
Common block /IO/		
IO5	-	Input unit (=5)
IO6	-	Output unit (=6)
IO7	-	Punch unit (=7)
Common block /BASIC/		
TITLE	(20)	Case title in A4
NASS	-	Number of assemblies
ANASS	-	Real equivalent of NASS
NADIAM	-	Number of assembly rows
NALPHA	-	Number of peripheral assemblies
ULOAD	-	Total core loading
HEIGHT	-	Total core height
BUDIF	-	Discharge burnup variation parameter
ISO	-	Iosotpic information generation option
W	-	Probability of absorption in one neighboring assembly for neutron born in an assembly
ALPHA	-	Albedo
ELIM	(2)	Minimum and maximum enrichments for a feed enrichment search
IHAL	-	Haling calculation option
POWER	-	Total core thermal power
DEBUG	-	(Logical) Debug option
Common block /COST/		
IF COST	-	Not used

Table A.2 (continued)

Variable	Dimension	Description
Common block /LOOP/		
NA	-	Number of values of ALPHA to be used in the sequence of cases
NW	-	Number of values of W to be used in the sequence of cases
DELA	-	Change in ALPHA between different values
DELW	-	Change in W between different values
Common block /COEF/		
COFIT	(11)	Not used
TEMP	(2)	Not used
BWRTH	-	Signifies if a "boron search" is to be performed at each burnup step
PWRTH	-	Not used
XEWRTH	-	Not used
Common block /TABLI/		
NBU	-	Number of burnups in the table
NRICH	-	Number of enrichments in the table
BURNUP	(20)	Burnups for tabular data in increasing order
ENRICH	(20)	Enrichments for tabular data in any order
ITYPE	(20)	Fuel type corresponding to each enrichment
BIAS	(20)	Axial bias factor as a function of burnup
BPWRTH	(20)	Worth of a burnable poison rod in one assembly as a function of burnup
CYBIAS	-	General bias factor
Common block /TABLE2/		
KINF	(20,20)	(Real) k_∞ data as a function of burnup for each enrichment

Table A.2 (continued)

Variable	Dimension	Description
ISOTOP	(20,20,6)	(Real) Isotopic data as a function of burnup for each enrichment and for different sets of isotopic parameters
POWFIS	(20)	Power (κ) to fission (ν) ratio for each enrichment
Common Block /CYCL1/		
C1RICH	(10)	Enrichments for each of the first cycle regions
C1TYPE	(10)	(Integer) Fuel type for each region in cycle 1
C1ID	(10)	Individual assembly loading for each region in cycle 1
NAC1	(10)	Number of assemblies in each region in cycle 1
Common Block /FEED/		
NAFEED	(21,3)	The number of feed assemblies in each cycle for each feed number
FTYPE	(21,3)	(Integer) Type of fuel for each cycle and each feed
FLOAD	(21,3)	The assembly loading for each cycle for each feed
FRICH	(21,3)	The feed enrichment for each cycle and each feed
FID	(21,3)	The feed alphanumeric label (A4) for each cycle and each feed
Common Block /CYDAT/		
ISURCH	(21)	Search parameter for each cycle
ISHUF	(21)	Shuffle generation option for each cycle
BP	(21)	Number of burnable poison rods for each cycle
CYBU	(21)	Cycle burnup for each cycle

Table A.2 (continued)

Variable	Dimension	Description
EOLK	(21)	End of cycle k_{eff} for each cycle
NFEED	(21)	Number of feed regions in each cycle
NREG	(21)	Number of regions in each cycle
CYLOAD	(21)	Total core loading for each cycle
CAPFAC	(21)	Capacity factor for each cycle
CYTIM	(21)	Elapsed cycle time for each cycle
SDLEN	(21)	Refueling shutdown length for each cycle
IFEDGE	(21)	Edge data generation option for each cycle
Common block /RGDAT/		
NEDGE	(2,1000)	Edge data packed according to IEDGE. The first number is of the form 100* I + J where I, J are the adjoining regions and the second number is the number of common edges
IEDGE	(21)	Location of last data item in NEDGE for each cycle. Data for Cycle I is between IEDGE (I-1) + 1 and IEDGE (I)
NAREG	(15,21)	Number of assemblies in each region in each cycle
ISHUF	(15,21)	Shuffle parameter for each region in each cycle. It gives the region and cycle number from which those assemblies came from
BPFRAC	(15,21)	Fraction of burnable poisons in each region in each cycle
BOCBU	(15,21)	Beginning of cycle burnup in each region and in each cycle
EOCBU	(15,21)	End of cycle burnup in each region and in each cycle
IDXREG	(15,21)	Index for each region and each cycle specifying the cycle (K) and feed number (J) in that cycle that was the original source of this region. The number is of the form 100 K + J.

Table A.2 (continued)

Variable	Dimension	Description
NADIS	(15,21)	The number of assemblies discharged from each region in each cycle
Common Block /CYDEP/		
BSHARE	(15,10)	The power sharings in each region at the different depletion steps for a particular cycle
BBU	(10)	The core burnups for the depletion steps for the cycle
BKEFF	(10)	The core k_{eff} at each depletion step for the cycle
NBBU	-	The number of depletion steps
Common Block /EDGE/		
IDGE	(15,20)	Storage for edge data in a full matrix for a single cycle

APPENDIX B

PROGRAM LISTING

On the following pages is the listing of the FLAC computer code for IBM computers. Change cards for modification of the program to enable running on a CDC computer are included as comments with CDC in columns 1-3.

```
CDC   PROGRAM FLAC(INPUT,OUTPUT,PUNCH,TAPE5=INPUT,TAPE6=OUTPUT,
CDC  1      TAPE7=PUNCH)
C
C     PROGRAM FLAC - FLARE BASED CYCLING PROGRAM
C           WRITTEN BY CHARLES L. BEARD, JR. SPRING 1978
C           NUCLEAR ENGINEERING DEPARTMENT, MIT
C           NOW WITH WESTINGHOUSE NUCLEAR FUEL DIVISION
C           PARTIAL FUNDING BY YANKEE ATOMIC ELECTRIC CO.
C
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
      COMMON /LOOP/ NA,NW,DELA,DELW
C
   10 CONTINUE
      CALL READIN
      WW=W
      DO 20 I=1,NA
```

```
      W=WW                                                              MAIN0037
      DO 15 J=1,NW                                                      MAIN0038
      CALL PREP                                                         MAIN0039
      CALL INEDIT                                                       MAIN0040
      CALL CALC                                                         MAIN0041
      CALL EQCALC                                                       MAIN0042
      CALL RGEDIT                                                       MAIN0043
      W=W+DELW                                                          MAIN0044
   15 CONTINUE                                                          MAIN0045
      ALPHA=ALPHA+DELA                                                  MAIN0046
   20 CONTINUE                                                          MAIN0047
      GO TO 10                                                          MAIN0048
      END                                                               MAIN0049
```

```
      BLOCK DATA
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
C
      DATA BURNUP / 0., 4., 8., 12., 16., 20., 24., 28., 32., 36., 40.,
     1             44., 48., 7*0. /
      DATA NBU,NRICH / 13,0 /
      DATA BPWRTH / 20*0.0 /
      DATA ISURCH,ISHUF,NEDGE,IEDGE / 42*0, 2021*0 /
      DATA IFEDGE / 21*0 /
      DATA NAC1/ 10*0 /
      DATA COFIT,TEMP,BWRTH,DPWRTH,XEWRTH / 16*0. /
      DATA BPFRAC / 315*0.0 /
      DATA ISOTOP / 2400*0.0 /
      DATA IO5,IO6,IO7 / 5,6,7 /
      END
```

```
      SUBROUTINE HEAD
C
      COMMON /IO/ IO5,IO6,IO7
      COMMON /BASIC/ TITLE(20)
C
      DATA NP / 0 /
C
      NP=NP+1
      WRITE (IO6,101)
      WRITE (IO6,102) TITLE,NP
      RETURN
C
  101 FORMAT(1H1,48X,14H**** FLAC ****  )
  102 FORMAT(1H0,14X,20A4,10X,4HPAGE,I3 //)
C
      END
```

```
      SUBROUTINE BUGOUT(NAME,ARRAY,I1,I2,I3,J1,J2,J3,IT)
C
      COMMON /IO/ IO5,IO6,IO7
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      DIMENSION ARRAY(I3,J3),NAME(2)
C
      IF (.NOT.DEBUG) GO TO 100
      IF (I3.GT.1.OR.J3.GT.1)  GO TO 10
      IF (IT.EQ.2) GO TO 5
      WRITE (IO6,101) NAME,ARRAY(1,1)
      GO TO 100
    5 CONTINUE
      WRITE (IO6,102) NAME,ARRAY(1,1)
      GO TO 100
   10 CONTINUE
      WRITE (IO6,103)
      DO 30 J=J1,J2
      DO 25 I=I1,I2,8
      II=MIN0(I2,I+7)
      IF (IT.EQ.2) GO TO 15
      WRITE (IO6,104) NAME,I,J,(ARRAY(K,J),K=I,II)
      GO TO 25
   15 CONTINUE
      WRITE (IO6,105) NAME,I,J,(ARRAY(K,J),K=I,II)
   25 CONTINUE
   30 CONTINUE
  100 RETURN
C
  101 FORMAT(1H0,9X,A4,A2,1H=,I12)
  102 FORMAT(1H0,9X,A4,A2,1H=,E12.5)
  103 FORMAT(1X)
  104 FORMAT(10X,A4,A2,1H(,I3,1H,,I3,2H)=,8I12)
  105 FORMAT(10X,A4,A2,1H(,I3,1H,,I3,2H)=,8E12.5)
C
```

END

BUG00037

```
      FUNCTION TERP(B,E,IT,XX,IOPT,KB ,BPK)
C
C     FUNCTION TERP -- INTERPOLATION FUNCTION FOR ARRAYS BASED ON
C           BURNUP AND ENRICH.  DOUBLE INTERPOLATION UNING 3 POINT FIT.
C           BURNUP FIT IS ONE POINT BELOW AND TWO ABOVE IF POSSIBLE.
C           ENRICH FIR IS ONE POINT BELOW, ONE POINT ABOVE AND NEXT
C           CLOSEST POINT IF THIS IS POSSIBLE.  ENRICH FIT WILL AUTO-
C           DEGENERATE TO LINEAR IF EXTRAPOLATION REQUIRED.
C
C           VARIALBES IN CALL STATEMENT ARE -
C           B     BURNUP AT WHICH DATA IS WANTED
C           E     ENRICHMENT AT WHICH DATA IS WANTED
C           IT    TYPE OF FUEL - INTERPOLATION USES ONLY THAT FUEL TYPE
C           XX(20,20)  ARRAY TO BE INTERPOLATED - FIRST INDEX IS
C                 BURNUP, SECOND INDEX IS ENRICH
C           IOPT  OPTION, IF =1, THEN JUST CALLED WITH A NEW ARRAY -
C                 POINT AND INTERPOLATION CONSTANTS SAME AS LAST CALL
C           IF =2, THEN INTERPOLATE ONLY ON ENRICHMENT.
C           KB     (REAL) K BIAS VALUE
C           BPK   BP WOTTH PER ROD AT BURNUP BU
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
```

```
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),      TERP0037
     1              FID(21,3)                                               TERP0038
      INTEGER FTYPE,C1TYPE                                                  TERP0039
      COMMON /COST / IFCOST                                                 TERP0040
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH                  TERP0041
      REAL KB                                                               TERP0042
      DIMENSION XX(20,20),EMIN(3),INDEX(3)                                  TERP0043
      DATA IERR / 0 /                                                       TERP0044
C                                                                           TERP0045
      IF (IOPT.EQ.1) GO TO 65                                               TERP0046
      IF (IOPT.EQ.2) GO TO 12                                               TERP0047
      I=NBU                                                                 TERP0048
      J=0                                                                   TERP0049
      DO 10 K=2,NBU                                                         TERP0050
      IF (B.GT.BURNUP(K-1).AND.B.LE.BURNUP(K)) I=K                          TERP0051
   10 CONTINUE                                                              TERP0052
      IF (B.LE.BURNUP(1)) I=2                                               TERP0053
      I=MINO(I,NBU-1)                                                       TERP0054
      I1=I-1                                                                TERP0055
      I2=I                                                                  TERP0056
      I3=I+1                                                                TERP0057
      B1=(BURNUP(I2)-B)*(BURNUP(I3)-B) / (BURNUP(I2)-BURNUP(I1))/           TERP0058
     1         (BURNUP(I3)-BURNUP(I1))                                      TERP0059
      B2=(BURNUP(I1)-B)*(BURNUP(I3)-B) / (BURNUP(I1)-BURNUP(I2))/           TERP0060
     2         (BURNUP(I3)-BURNUP(I2))                                      TERP0061
      B3=(BURNUP(I1)-B)*(BURNUP(I2)-B) / (BURNUP(I1)-BURNUP(I3))/           TERP0062
     3         (BURNUP(I2)-BURNUP(I3))                                      TERP0063
      KB=B1*BIAS(I1)+B2*BIAS(I2)+B3*BIAS(I3)                                TERP0064
      BPK=1.E-5*(B1*BPWRTH(I1)+B2*BPWRTH(I2)+B3*BPWRTH(I3))                 TERP0065
      KB=KB*CYBIAS                                                          TERP0066
C                                                                           TERP0067
   12 CONTINUE                                                              TERP0068
      EMIN(2)=100.                                                          TERP0069
      EMIN(3)=100.                                                          TERP0070
      EMIN(1)=-100.                                                         TERP0071
      INDEX(1)=0                                                            TERP0072
```

```
      INDEX(2)=0                                                        TERP0073
      INDEX(3)=0                                                        TERP0074
      N=0                                                               TERP0075
      DO 30 K=1,NRICH                                                   TERP0076
      IF (ITYPE(K).NE.IT) GO TO 30                                      TERP0077
      ERR=ENRICH(K)-E                                                   TERP0078
      IF (ERR.LT.0.0) GO TO 15                                          TERP0079
      IF (ERR.EQ.0.0) GO TO 20                                          TERP0080
      IF (EMIN(2).LT.ERR) GO TO20                                       TERP0081
      IF (EMIN(2).EQ.ERR) GO TO 30                                      TERP0082
      ER=ABS(EMIN(2))                                                   TERP0083
      KK=INDEX(2)                                                       TERP0084
      EMIN(2)=ERR                                                       TERP0085
      INDEX(2)=K                                                        TERP0086
      GO TO 25                                                          TERP0087
   15 CONTINUE                                                          TERP0088
      IF (EMIN(1).GT.ERR) GO TO 20                                      TERP0089
      IF (EMIN(1).EQ.ERR) GO TO 30                                      TERP0090
      ER =ABS(EMIN(1))                                                  TERP0091
      KK=INDEX(1)                                                       TERP0092
      EMIN(1)=ERR                                                       TERP0093
      INDEX(1)=K                                                        TERP0094
      GO TO 25                                                          TERP0095
   20 CONTINUE                                                          TERP0096
      ER=ABS(ERR)                                                       TERP0097
      KK=K                                                              TERP0098
   25 IF (ER.GE.EMIN(3)) GO TO 30                                       TERP0099
      EMIN(3)=ER                                                        TERP0100
      INDEX(3)=KK                                                       TERP0101
   30 CONTINUE                                                          TERP0102
      N=0                                                               TERP0103
      DO 35 K=1,3                                                       TERP0104
      J=INDEX(K)                                                        TERP0105
      IF (ABS(EMIN(K)).EQ.100.) EMIN(K)=100.                            TERP0106
      IF (EMIN(K).NE.100.) EMIN(K)=ENRICH(J)                            TERP0107
      IF (EMIN(K).NE.100.) N=N+1                                        TERP0108
```

```
35 CONTINUE
   DO 40 K=1,2
   K1=K+1
   DO 40 J=K1,3
   IF (EMIN(J).GT.EMIN(K)) GO TO 40
   EE=EMIN(K)
   EMIN(K)=EMIN(J)
   EMIN(J)=EE
   J1=INDEX(K)
   INDEX(K)=INDEX(J)
   INDEX(J)=J1
40 CONTINUE
   IF (N.EQ.0) WRITE (IO6,102) IT
   IF (N.EQ.0) IERR=IERR+1
   IF (N.EQ.0) GO TO 70
   GO TO (45,50,60), N
45 CONTINUE
   J1=INDEX(1)
   J2=INDEX(1)
   J3=INDEX(1)
   E1=1.0
   E2=0.0
   E3=0.0
   GO TO 65
50 CONTINUE
   J1=INDEX(1)
   J2=INDEX(2)
   J3=INDEX(2)
   E3=0.0
   E2=(ENRICH(J1)-E)/(ENRICH(J1)-ENRICH(J2))
   E1=1.0-E2
   GO TO 65
60 CONTINUE
   J1=INDEX(1)
   J2=INDEX(2)
   J3=INDEX(3)
```

```
      E1=(ENRICH(J2)-E)*(ENRICH(J3)-E)/ (ENRICH(J2)-ENRICH(J1)) /
     1       (ENRICH(J3)-ENRICH(J1))
      E2=(ENRICH(J1)-E)*(ENRICH(J3)-E)/ (ENRICH(J1)-ENRICH(J2)) /
     2       (ENRICH(J3)-ENRICH(J2))
      E3=(ENRICH(J1)-E)*(ENRICH(J2)-E)/ (ENRICH(J1)-ENRICH(J3)) /
     3       (ENRICH(J2)-ENRICH(J3))
   65 CONTINUE
C
      IF  (IOPT.NE.2) GO TO 66
      TERP= E1*XX(J1,1)  + E2*XX(J2,1)  + E3*XX(J3,1)
      GO TO 70
C
   66 CONTINUE
      X1= B1*XX(I1,J1)  + B2*XX(I2,J1)  + B3*XX(I3,J1)
      X2= B1*XX(I1,J2)  + B2*XX(I2,J2)  + B3*XX(I3,J2)
      X3= B1*XX(I1,J3)  + B2*XX(I2,J3)  + B3*XX(I3,J3)
      TERP=E1*X1+E2*X2+E3*X3
   70 CONTINUE
      IF (DEBUG) WRITE (IO6,101) B,E,IT,TERP
      IF (IERR.GT.10) CALL EXIT
      RETURN
C
  101 FORMAT(10X,11HTERP AT BU= ,F9.3,3H E=,F6.2,7H, TYPE=,I2,5H, IS ,
     1       E12.5)
  102 FORMAT(24H0ERROR IN TERP FOR TYPE  ,I3)
C
      END
```

```
      SUBROUTINE READIN
C
C     SUBROUTINE READIN - READS IN ALL INPUT DATA
C
C                           INPUT DATA
C
C
C     THE FOLLOWING INPUT FORMAT IS USED.  FOR MULTIPLE CASES STACK THE
C           DECKS.  THE PROGRAM WILL CHECK FOR END OF DATA.
C
C           INPUT PARAMETERS IN ORDER ARE:
C
C     COLS. PARAM. TYPE SUBPT  UNITS       DESCRIPTION
C
C
C     ****  TITLE CARD  (20A4)
C
C      1-80 TITLE   A    (20)     --  FULL CARD OF TITLE DATA FOR CASE
C
C     ****  CASE DATA CARD A  (24I3)
C
C      1- 3 NASS    I    --     --    NUMBER OF ASSEMBLIES IN WHOLE CORE
C
C      4- 6 NADIAM  I    --     --    NUMBER OF ASSEMBLIES ACROSS CORE
C
C      7- 9 NBU     I    --     --    NUMBER OF BURNUPS IN INTERPOLATION
C                                     TABLE. IF NOT ENTERED, USE DEFAULT
C                                     OR PREVIOUS CASE BURNUPS.
C                                     IF =-1, THEN TABLES INPUT AS
C                                     COEFFICIENTS OF DEFINED FUNCTIONS
C                                     AS IN SIMULATE.
C
C     10-12 NRICH  I     --     --    NUMBER OF K INF. SETS TO BE READ
C                                     IN.  IF =0, USE DEFAULT OR PREVIOUS
C                                     CASE DATA, IF >0, REPLACE PREVIOUS
C                                     SET WITH NEW SET.  IF <0, ADD THIS
```

```
C                                                MANY NEW SETS TO PREVIOUS SETS.          READ0037
C                                                                                         READ0038
C      13-15 ISO       I     --      --          IF =1, READ IN ISOTOPIC DATA ALONG       READ0039
C                                                WITH K INF. DATA.                        READ0040
C                                                IF =-1, GENERATE ISOTOPIC DATA FROM      READ0041
C                                                MUDDLE FIT.                              READ0042
C                                                                                         READ0043
C      16-18 IFCOST    I     --      --          IF =1, PREPARE DATA SET FOR A FUEL       READ0044
C                                                CYCLE COST CALCULATION.                  READ0045
C                                                                                         READ0046
C      19-21 INFIT     I     --      --          IF =1, READ IN FIT DATA.                 READ0047
C                                                                                         READ0048
C      22-24 IHAL      I     --      --          IF =1, A HALING CALCULATION WILL BE      READ0049
C                                                DONE FOR EACH CYCLE.                     READ0050
C                                                                                         READ0051
C      25-27 DEBUG     L     --      --          IF =1, DEBUG PRINT OPTION                READ0052
C                                                                                         READ0053
C      28-30 NW        I     --      --          NUMBER OF DIFFERENT VALUES OF W TO       READ0054
C                                                USE. IF NOT ENTERED, SET TO 1.           READ0055
C                                                                                         READ0056
C      31-33 NA        I     --      --          NUMBER OF DIFFERENT VALUES OF ALPHA      READ0057
C                                                TO USE.  IF NOT ENTERED, SET TO 1.       READ0058
C                                                                                         READ0059
C                                                                                         READ0060
C      ****  CASE DATA CARD B  (12F6.2)                                                   READ0061
C                                                                                         READ0062
C      1- 6 W          R     --      --          PROBABILITY THAT A NEUTRON BORN IN       READ0063
C                                                AN ASSEMBLY WILL BE ABSORBED IN AN       READ0064
C                                                ADJACENT ASSEMBLY.  DEFAULT = 0.115.     READ0065
C                                                                                         READ0066
C       7-12 ALPHA     R     --      --          ALBEDO ON EXTERIOR EDGE OF CORE.         READ0067
C                                                DEFAULT = 0.30.                          READ0068
C                                                                                         READ0069
C      13-18 ELIM      R     (1)    W/O          MINIMUM FEED ENRICHMENT IN SEARCH        READ0070
C                                                                                         READ0071
C      19-24 ELIM      R     (2)    W/O          MAXIMUM FEED ENRICHMENT IN SEARCH        READ0072
```

```
C                                         IF NOT INPUT, SET TO 4.5
C
C     25-30 BUDIF    R     --   GWD/MTM BURNUP DIFFERENCE FOR SPLIT REGION
C                                       BURNUP OF SPLIT REGION IS GIVEN BY
C                                        BU = AVG.BU - BUDIF*(1-N KEPT)/N)
C
C     31-36 CYBIAS   R     --      --   K EFF BIAS FACTOR - DIVIDES INTO K
C
C     37-42 HEIGHT   R     --     FEET  CORE HEIGHT FOR GENERATION OF AXIAL
C                                       BIAS FACTOR AS A FCN OF BURNUP.
C
C     43-48 ULOAD    R     --     MTM   TOTAL CORE AVERAGE LOADING.
C
C     49-54 POWER    R     --     MW T  TOTAL CORE POWER.
C
C     55-60 DELW     R     --      --   INCREMENTAL CHANGE IN W TO BE USED
C                                       TO GENERATE NW-1 VALUES OF W AFTER
C                                       THE BASE INPUT VALUE.
C
C     61-66 DELA     R     --      --   INCREMENTAL CHANGE IN ALPHA TO BE
C                                       USED TO GENERATE NA-1 VALUES OF
C                                       ALPHA AFTER THE BASE INPUT VALUE.
C
C
C     ****  BURNUP DATA CARDS  (12F6.2)          (ONLY IF NBU>0)
C
C           BURNUP   R    (I) GWD/MTM BURNUP ARRAY IN ORDER OF INCREASING
C                                       BURNUP. 12 ITEMS PER CARD, UP TO 20
C                                       ENTRIES READ IN.  ONLY READ IN IF
C                                       NBU>0, THEN NBU ENTRIES READ
C
C
C     ****  AXIAL BIAS CARDS  (6E12.5)           (ONLY IF NBU>0, HEIGHT=0)
C
C           BIAS     R    (I)      --   AXIAL BIAS FACTOR AS A FUNCTION BU,
C                                       ENTERED ONLY IF NBU>0 AND HEIGHT=0.
```

```
C                                        NBU ENTRIES CORRESPONDING TO
C                                        BURNUP(I), 6 ENTRIES PER CARD.
C
C
C     ****  BP WORTH CARDS      (6E12.5)          (ONLY IF NBU>0)
C
C           BP         R      (I)     PCM   WORTH OF ONE BP ROD/ ASSEMBLY AS A
C                                        FUNCTION OF BURNUP(I). ENTERED ONLY
C                                        IF NBU>0. NBU ENTRIES, 6 PER CARD.
C
C
C     ****  BP COEFFICIENT CARD (6E12.5)          (ONLY IF NBU<0)
C
C           B          R      (I)     --    COEFFICIENTS OF FUNCTION FOR WORTH
C                                        OF ONE BP ROD/ASSEMBLY. FUNCTION IS
C                                        B(1)*(1.0 + B(2)*BU + B(3)*BU**2 +
C                                          B(4)*BU**3 + B(5)*BU**5)
C                                        WHERE BU IS THE BURNUP IN GWD/MTU.
C
C
C     ****  ENRICHMENT CARDS    (12F6.2)          (ONLY IF NRICH NE 0)
C
C           ENRICH     R      (I)     W/O   ENRICHMENT CORRESPONDING TO EACH
C                                        DATA SET. W/O U-235 OR W/O PU.
C                                        NRICH VALUES ENTERED, 12 PER CARD.
C
C
C     ****  FUEL TYPE CARDS     (12I6)            (ONLY IF NRICH NE 0)
C
C           ITYPE      I      (I)     --    FUEL TYPE CORRESPONDING TO EACH
C                                        DATA SET.  NRICH VALUES ENTERED 12
C                                        PER CARD.
C
C
C     ****  POWER/FISSION RATIO CARDS  (12F6.0) (ONLY IF NRICH NE 0)
C
```

```
C         POWFIS  R     (I)    MEV/N POWER TO FISSION SOURCE RATIO FOR
C                                    EACH FUEL TYPE. (KAPPA/NU) NRICH
C                                    VALUES, 12 PER CARD.
C
C
C
C   ****  K INF CARDS        (6E12.5)     (ONLY IF NRICH NE 0, NBU GE 0)
C
C         KINF    R     (I,J)   --   K INF CORRESPONDING TO BURNUP(I)
C                                    FOR ENRICHMENT J. NBU VALUES FOR
C                                    EACH SET, 6 VALUES PER CARD, NRICH
C                                    SETS.
C
C
C   ****  KINF COEFFICIENT CARDS (6E12.5) (ONLY IF NRICH NE 0, NBU<0)
C
C         B       R     (I)     --   COEFFICIENTS OF FUNCTION FOR K INF.
C                                    AS A FUNCTION OF BURNUP.  ONE SET
C                                    ENTERED FOR EACH ENRICHMENT.
C                                    THE FUNCTION IS OF THE FORM:
C                                    K INF = B(1)*Z1*Z2
C                                    Z1=1.0-B(2) - B(3)*(1.0+B(4)*BU)
C                                    Z2=1.0 - B(5)*BU + B(6)*BU**2 +
C                                      B(7)*BU**3 + B(8)*BU**4
C                                    WHERE BU IS BURNUP IN GWD/MTU.
C                                    NOTE THAT B(1) TO B(8) ARE TO BE
C                                    ENTERED, SO 2 CARDS/SET ARE NEEDED.
C
C
C   ****  ISOTOPIC DATA CARDS  (6E12.5)  (ONLY IF ISO=1, NRICH NE 0)
C
C         ISOTOP  R   (I,J,1) KG/MTM URANIUM CONTENT AS A FUNCTION OF
C                                    BURNUP(I).  NBU ENTRIES, 6 ENTRIES
C                                    PER CARD FOR ENRICH(J).
C
C         ISOTOP  R   (I,J,2) KG/MTM U-235 CONTENT AS A FUNCTION OF
```

```
C                                   BURNUP(I).  NBU ENTRIES, 6 ENTRIES           READ0181
C                                   PER CARD FOR ENRICH(J).                      READ0182
C                                                                                READ0183
C       ISOTOP  R  (I,J,3) KG/MTM   TOTAL PU CONTENT AS A FUNCTION OF            READ0184
C                                   BURNUP(I).  NBU ENTRIES, 6 ENTRIES           READ0185
C                                   PER CARD FOR ENRICH(J).                      READ0186
C                                                                                READ0187
C       ISOTOP  R  (I,J,4)    --    FISSILE/(TOTAL PU) RATIO AS A                READ0188
C                                   FUNCTION OF BURNUP(I) FOR ENRICH(J).         READ0189
C                                                                                READ0190
C          NOTE ***  THE ORDER OF CARDS ARE ALL BURNUPS IN SEQUENCE              READ0191
C                   ON CARD(S) TO FORM A SET.  THEN EACH SET FOR                 READ0192
C                   DIFFERENT ISOTOPIC INFORMATION (4 SETS) FOR                  READ0193
C                   ENRICH(J) TO FORM A LARGER SET WHICH IS THEN REPEATED        READ0194
C                   FOR EACH ENRICHMENT. IF NOT ENTERED, DEFAULT                 READ0195
C                   DATA WILL BE USED FROM FIT.  DATA IS ONLY USED               READ0196
C                   FOR COST DATA SET PREPARATION.                               READ0197
C                                                                                READ0198
C                                                                                READ0199
C    ****  COEFFICIENT FIT DATA SET  (5F6.2/ (6E12.5)/)                          READ0200
C                                             (ONLY IF INFIT =1)                 READ0201
C                                                                                READ0202
C     1- 6 XEWRTH  R     --      PCM  TOTAL HFP XENON WORTH AT BOL               READ0203
C                                                                                READ0204
C     7-12 DPWRTH  R     --      PCM  TOTAL DOPPLER DEFECT HFP TO HZP            READ0205
C                                     AT BOL.                                    READ0206
C                                                                                READ0207
C    13-18 BWRTH   R     --  PCM/PPM  BORON WORTH AT BOL.                        READ0208
C                                                                                READ0209
C    19-24 TEMP    R     (1)   DEG F  HZP TEMPERATURE.                           READ0210
C                                                                                READ0211
C    25-30 TEMP    R     (2)   DEG F  AVERAGE CORE HFP TEMPERATURE.              READ0212
C                                                                                READ0213
C          COFIT   R     (I)    --    MODERATOR TEMPERATURE FIT PARAMETERS       READ0214
C                                     UP TO 10 ALLOWED, 6 PER CARD.              READ0215
C                                                                                READ0216
```

```
C                                                                          READ0217
C     ****  CYCLE DATA **** ALL OF THE REMAINING DATA IS INPUT FOR EACH    READ0218
C                             CYCLE, THEN FOR THE NEXT CYCLE, ETC.  THE    READ0219
C                             CYCLES SHOULD BE IN INCREASING ORDER.        READ0220
C                             ONE MUST START WITH CYCLE 1, BUT BOC         READ0221
C                             BURNUPS MAY BE ENTERED.  NO CYCLE MAY BE     READ0222
C                             ELIMINATED FROM THE SEQUENCE, EXCEPT AT      READ0223
C                             THE END.  THUS ANY NUMBER OF CYCLES UP TO    READ0224
C                             20 MAY BE RUN AND THEY MUST BE CONSECUTIVE.  READ0225
C                             CYCLE 21 IS AN EQUILIBRIUM CYCLE AND IS      READ0226
C                             SEPARATE FROM THE REST.  THE CYCLE DATA      READ0227
C                             MUST BE FOLLOWED BY A BLANK CARD AFTER THE   READ0228
C                             LAST CYCLE INPUT.                            READ0229
C                                                                          READ0230
C                                                                          READ0231
C     ****  CYCLE OPTION CARD  (8I3,6F6.0)                                 READ0232
C                                                                          READ0233
C      1- 3 IS       I     --     --     CYCLE NUMBER 1-20 OR 21 FOR EQUIL.  READ0234
C                                        CYCLE.                            READ0235
C                                                                          READ0236
C      4- 6 ISURCH   I    (IS)    --     SEARCH PARAMETER FOR CYCLE IS.    READ0237
C                                        =1 SEARCH ON CYCLE BURNUP         READ0238
C                                        =2 SEARCH ON NUMBER OF BPS        READ0239
C                                        =3 DETERMINE EOL K EFF.           READ0240
C                                        =10+J SEARCH ON ENRICHMENT FOR    READ0241
C                                        FEED J IN CYCLE IS                READ0242
C                                                                          READ0243
C      7- 9 ISHUF    I    (IS)    --     SHUFFLE GENERATION OPTION         READ0244
C                                        =0 INPUT SHUFFLE FOR THIS CYCLE   READ0245
C                                        =-1 USE SHUFFLE FROM LAST CASE FOR  READ0246
C                                        THIS CYCLE.                       READ0247
C                                        >0 GENERATE SHUFFLE DATA AUTOMATICALLY  READ0248
C                                        =1 ASSUME PERITHERAL FEED AND CENTER  READ0249
C                                        ASSEMBLY FROM PREVIOUS CYCLE.     READ0250
C                                        =2 ASSUME PERIPHERAL FEED AND CENTER  READ0251
C                                        ASSEMBLY FROM CYCLE 1.            READ0252
```

```
C                                                                       READ0253
C     10-12 NREG     I    (IS)     --     NUMBER OF REGIONS IN CYCLE  (<16)  READ0254
C                                                                       READ0255
C     13-15 NFEED    I    (IS)     --     NUMBER OF FEED REGIONS IN CYCLE.   READ0256
C                                         MAY BE 0,1,2, OR 3.           READ0257
C                                                                       READ0258
C     16-18 IBOC     I     --      --     =1 IF BOC REGION BURNUPS ARE TO BE READ0259
C                                         ENTERED FOR THIS CYCLE.       READ0260
C                                         =0 IF BOC REGION BURNUPS ARE TO BE READ0261
C                                         CALCULATED USING SHUFFLE DATA AND  READ0262
C                                         EOC REGION BURNUPS FROM PREVIOUS   READ0263
C                                         CYCLES.                       READ0264
C                                         =-1 IF PREVIOUS CASE BOC REGION    READ0265
C                                         BURNUPS ARE TO BE USED THIS CYCLE. READ0266
C                                                                       READ0267
C     19-21 IFEDGE   I    (IS)     --     =0 INPUT THE EDGE DATA.       READ0268
C                                         =-1 USE THE EDGE DATA FROM THE     READ0269
C                                         PREVIOUS CASE FOR THIS CYCLE.      READ0270
C                                         >0 GENERATE EDGE DATA AUTOMATICALLY READ0271
C                                         =1 ASSUME A RING GEOMETRY IN THE   READ0272
C                                         CORE FOR THE EDGE DATA.       READ0273
C                                         =2 ASSUME A CHECKERBOARD GEOMETRY  READ0274
C                                         FOR THE INTERIOR OF THE CORE.      READ0275
C                                                                       READ0276
C     22-24 DUMMY    I                                                  READ0277
C                                                                       READ0278
C     25-30 CYBU     R    (IS) GWD/MTM DESIRED (OR GUESSED) CYCLE BURNUP.    READ0279
C                                                                       READ0280
C     31-36 BP       R    (IS)     --     NUMBER OF BP RODS (OR GUESS) IN    READ0281
C                                         CORE.                         READ0282
C                                                                       READ0283
C     37-42 EOLK     R    (IS)     --     DESIRED END OF LIFE K EFF FOR CYCLE. READ0284
C                                         CAN BE VARIED TO CHANGE FOR EOL    READ0285
C                                         BORON CONCENTRATION, COASTDOWN OR  READ0286
C                                         EARLY SHUTDOWN.               READ0287
C                                                                       READ0288
```

```
C      43-48 CAPFAC   R    (IS) PERCENT CAPACITY FACTOR FOR CYCLE, EXCLUDING       READ0289
C                                       REFUELING SHUTDOWN.                        READ0290
C                                                                                  READ0291
C      49-54 CYTIM    R    (IS)    DAYS CYCLE LENGTH, EXCLUDING SHUTDOWN.          READ0292
C                                       ONLY 2 OF 3 (CYBU, CAPFAC, CYTIM)          READ0293
C                                       NEED BE ENTERED.  IF 3 ENTERED,            READ0294
C                                       CYTIM IS RECALCULATED.                     READ0295
C                                                                                  READ0296
C      55-60 SDLEN    R    (IS)    DAYS REFUELING SHUTDOWN LENGTH.                 READ0297
C                                       IF NOT ENTERED, SET TO 42.                 READ0298
C                                                                                  READ0299
C                                                                                  READ0300
C      ****  FEED DATA CARDS  (2I3,A3,I3,2F6.0)                                    READ0301
C                          INPUT FOR EACH REGION IN CYCLE 1 OR JUST THE FEED       READ0302
C                          REGIONS FOR OTHER CYCLES.  START WITH THE LOWEST        READ0303
C                          ENRICHMENT REGION AND PUT IN ORDER OF INCREASING        READ0304
C                          ENRICHMENT.                                             READ0305
C                                                                                  READ0306
C       1- 3 NAFEED   I    (IS,I)  --   NUMBER OF FEED I (OR REGION I IN           READ0307
C            NAC1     I     (I)        CYCLE 1) ASSEMBLIES.                        READ0308
C                                                                                  READ0309
C       4- 6 FTYPE    I    (IS,I)  --   FUEL TYPE FOR FEED I (OR REGION I          READ0310
C            C1TYPE   I     (I)    --   IN CYCLE 1) CORRESPONDING TO INPUT         READ0311
C                                       TABLE OF K INF.  INTERPOLATION WILL        READ0312
C                                       ONLY BE DONE WITH IDENTICAL FUEL           READ0313
C                                      TYPES.                                      READ0314
C                                                                                  READ0315
C       7- 9 FID      A    (IS,I)  --   ID FOR FEED REGION 2. (IE. 5A,6PU).        READ0316
C            C1ID     A     (I)    --   THIS IS USED FOR LABELING OF OUTPUT.       READ0317
C                                                                                  READ0318
C      10-12 DUMMY    I                                                            READ0319
C                                                                                  READ0320
C      13-18 FRICH    R    (IS,I) W/O   ENRICHMENT OF FEED I IN CYCLE IS OR        READ0321
C            C1RICH   R     (I)   W/O   ENRICHMENT OF REGION I IN CYCLE 1.         READ0322
C                                                                                  READ0323
C      19-24 FLOAD    R    (IS,I) MTM   LOADING PER ASSEMBLY FOR FEED I IN         READ0324
```

```
C        C1LOAD   R     (I)    MTM   CYCLE IS OR REGION I IN CYCLE 1.
C                                    IF NOT ENTERED, SET TO ULOAD/NASS.
C
C
C    ****  REGION SIZE CARD  (12I6)            (ONLY IF ISHUF(IS)=0)
C
C        NAREG    I     (I,IS)  --   NUMBER OF ASSEMBLIES IN EACH REGION
C                                    STARTING WITH THE HIGHEST BURNUP
C                                    (INNERMOST) REGION.  NREG(IS) ENTRIES,
C                                    12 PER CARD.
C
C
C    ****  SHUFFLE CARD  (12I6)                (ONLY IF ISHUF(IS)=0)
C
C        ISHUF    I     (I,IS)  --   SHUFFLE INICIES SPECIFYING PREVIOUS
C                                    LOCATION OF EACH REGION(I) IN CYCLE
C                                    IS.  VALUE OF FORM 100*K+L WHERE
C                                    K IS THE PREVIOUS CYCLE NUMBER AND
C                                    L IS THE REGION NUMBER IN THE CYCLE.
C                                    FOR A FEED REGION, K=0 AND L IS THE
C                                    FEED NUMBER (1,2, OR 3)  NREG(IS)
C                                    ENTRIES, 12 PER CARD.
C
C
C    ****  EDGE CARDS   (11I6)                 (ONLY IF IFEDGE(IS)=0)
C
C     1- 6 JR       I      --       --   REGION NUMBER CORRESPONDING TO DATA
C                                    ON THE WHOLE CARD.
C
C     7-12 NEDGE    I     (1,K)     --   SPECIFICATIONS OF THE NUMBER OF EDGES
C    13-18 NEDGE    I     (2,5)     --   ADJOINING REGION JR IN CYCLE IS.
C    19-24 NEDGE    I     (1,K+1)   --   NEDGE(1,.) IS THE REGION NUMBER
C    25-30 NEDGE    I     (2,K+1)   --   ADJOINING REGION JR AND NEDGE(2,.)
C          ETC.                      IS THE NUMBER OF EDGES BETWEEN THEM.
C                                    FOR EDGES ON THE EXTERIOR, SET
C                                    NEDGE(1,.)=0.  NOTE THAT K IS JUST
```

```
C                                           A COUNTER.  ALSO ONLY NON-ZERO
C                                           NUMBER OF EDGES MUST BE ENTERED AND
C                                           ONLY ONE SET FOR EACH REGION PAIR.
C                                           5 PAIRS MAY BE ENTERED ON EACH CARD.
C                                   NOTE:  LAST CARD WITH EDGE DATA SHOULD BE
C                                   FOLLOWED BY A BLANK CARD.
C
C
C     ****  BP FRACTION CARD  (12F6.2)           (ONLY IF BP(IS) NE 0)
C
C           BPFRAC  R   (I,IS)  --    FRACTION OF BP RODS IN EACH REGION
C                                     I IN CYCLE IS.  NREG(IS) ENTRIES,
C                                     12 ENTRIES PER CARD.
C
C
C     ****  BOC BURNUP CARD   (12F6.2)           (ONLY IF IBOC=1)
C
C           BOCBU    R   (I,IS)GWD/MTM BOC REGION BURNUPS FOR CYCLE IS.
C                                   NREG(IS) ENTRIES, 12 ENTRIES / CARD.
C
C
C     ****  END OF CYCLE (IS) DATA - REPEAT SET FOR NEXT CYCLE.
C     ****  AFTER LAST CYCLE DATA SET, PUT A BLANK CARD.
C     ****  CASES MAY BE STACKED DIRECTLY.
C
C
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1              BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1              BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
```

```
     1                NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2                SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/  NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1                BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2                IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/  C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED /  NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1                FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
      COMMON /LOOP/ NA,NW,DELA,DELW
      DIMENSION IDUM(24),DUM(12)
      DIMENSION B(10)
      DATA BLANK / 3H    /
C
      READ (IO5,901,END=200) TITLE
CDC   READ (IO5,901        ) TITLE
CDC   IF (EOF(IO5)) 200,5
CDC 5 CONTINUE
C
      READ (IO5,910) NASS,NADIAM,NB,NR,IS,IFCOST,INFIT,IHAL,DEBUG,NW,NA
C
      READ (IO5,904) W,ALPHA,ELIM(1),ELIM(2),BUDIF,CYBIAS,HEIGHT,
     1      ULOAD,POWER,DELW,DELA
      CALL BUGOUT(8HW       ,W,1,1,1,1,1,1,2)
      CALL BUGOUT(8HALPHA   ,ALPHA,1,1,1,1,1,1,2)
      ANASS=NASS
      ISO=IS
      IF (W.EQ.0.0) W=0.115
      IF (ALPHA.EQ.0.0) ALPHA=0.30
      IF (ULOAD.EQ.0.0) ULOAD=1.0
      IF (ELIM(2).EQ.0.0) ELIM(2)=4.5
      IF (POWER.EQ.0.0) POWER=1.0
      IF (CYBIAS.EQ.0.0) CYBIAS=1.0
      IF (NA.EQ.0) NA=1
```

```
      IF (NW.EQ.0) NW=1
C
      IF (NB.LE.0) GO TO 10
      READ (IO5,904) (BURNUP(I),I=1,NB)
      NBU=NB
      IF (HEIGHT.EQ.0.0) READ (IO5,905) (BIAS(I),I=1,NB)
      READ (IO5,905) (BPWRTH(I),I=1,NB)
   10 CONTINUE
C
      IF (NB.GE.0) GO TO 13
      READ (IO5,905) (B(I),I=1,5)
      DO 12 I=1,NBU
      BU=BURNUP(I)
      BPWRTH(I)=B(1)*(1.0+B(2)*BU+B(3)*BU**2+B(4)*BU**3+B(5)*BU**4)
      BPWRTH(I)=AMAX1(0.0,BPWRTH(I))
   12 CONTINUE
   13 CONTINUE
C
      IF (NR.EQ.0) GO TO 35
      IF (NR.LT.0) GO TO 15
      NR1=1
      NRICH=NR
      GO TO 20
   15 CONTINUE
      NR1=NRICH+1
      NRICH=NRICH-NR
   20 CONTINUE
      READ (IO5,904) (ENRICH(I),I=NR1,NRICH)
      READ (IO5,906) (ITYPE(I),I=NR1,NRICH)
      READ (IO5,904) (POWFIS(I),I=NR1,NRICH)
      DO 25 I=NR1,NRICH
C
      IF (NB.LT.0) GO TO 21
      READ (IO5,905) (KINF(J,I),J=1,NBU)
      GO TO 26
C
```

```
   21 CONTINUE
      READ (IO5,905) (B(J),J=1,8)
      DO 22 J=1,NBU
      BU=BURNUP(J)
      A=1.0-B(5)*BU+B(6)*BU**2+B(7)*BU**3+B(8)*BU**4
      KINF(J,I)=B(1)*A*(1.0-B(2)-B(3)*(1.0+B(4)*BU))
   22 CONTINUE
   26 CONTINUE
   25 CONTINUE
C
      IF (ISO.NE.1) GO TO 35
      DO 30 I=NR1,NRICH
      DO 30 L=1,6
      READ (IO5,905) (ISOTOP(J,I,L),J=1,NBU)
   30 CONTINUE
   35 CONTINUE
C
      DO 41 I=1,21
   41 ISURCH(I)=0
C
      IF (INFIT.EQ.0) GO TO 40
      READ (IO5,904) XEWRTH,DPWRTH,BWRTH,TEMP
      READ (IO5,907) COFIT
   40 CONTINUE
C
C       CYCLE INPUT ON A CYCLE BY  CYCLE BASIS
C
      READ (IO5,903,END=150) (IDUM(I),I=1,8),(DUM(I),I=1,6)
CDC   READ (IO5,903          ) (IDUM(I),I=1,8),(DUM(I),I=1,6)
CDC   IF (EOF(IO5)) 150,6
CDC 6 CONTINUE
      IF (IDUM(1).EQ.0) GO TO 150
      IS=IDUM(1)
      ISURCH(IS)=IDUM(2)
      ISHUF(IS)=IDUM(3)
      NREG(IS)=IDUM(4)
```

```
      NFEED(IS)=IDUM(5)
      IBOC=IDUM(6)
      IFEDGE(IS)=IDUM(7)
      CYBU(IS)=DUM(1)
      BP(IS)=DUM(2)
      EOLK(IS)=DUM(3)
      CAPFAC(IS)=DUM(4)
      CYTIM(IS)=DUM(5)
      SDLEN(IS)=DUM(6)
      IF (SDLEN(IS).EQ.0.0) SDLEN(IS)=42.
C
      IF (IS.GT.1) GO TO 50
      NR=NREG(IS)
      DO 45 I=1,NR
      READ (I05,909) NAC1(I),C1TYPE(I),C1ID(I),IDUM(1),C1RICH(I),
     1               C1LOAD(I)
      IF (C1LOAD(I).EQ.0.0) C1LOAD(I)=ULOAD/ANASS
C
CDC   IF (C1ID(I).EQ.BLANK) ENCODE (3,903,C1ID(I)) I
      NF=NR
   45 CONTINUE
      GO TO 60
C
   50 CONTINUE
      NR=NFEED(IS)
      IF (NR.EQ.0) GO TO 60
      DO 55 I=1,NR
      READ (I05,909) NAFEED(IS,I),FTYPE(IS,I),FID(IS,I),IDUM(1),
     1               FRICH(IS,I),FLOAD(IS,I)
      NF=NF+1
C
CDC   IF (FID(IS,I).EQ.BLANK) ENCODE (3,903,FID(IS,I)) NF
      IF (FLOAD(IS,I).EQ.0.0) FLOAD(IS,I)=ULOAD/ANASS
   55 CONTINUE
   60 CONTINUE
C
```

```
      NR=NREG(IS)                                                       READ0541
      IF (ISHUF(IS).NE.0) GO TO 65                                      READ0542
      IF (IS.EQ.1) GO TO 65                                             READ0543
      READ (IO5,906) (NAREG(I,IS),I=1,NR)                               READ0544
      READ (IO5,906) (JSHUF(I,IS),I=1,NR)                               READ0545
   65 CONTINUE                                                          READ0546
C                                                                       READ0547
      IN=0                                                              READ0548
      IF (IS.GT.1) IN=IEDGE(IS-1)                                       READ0549
      IN2=IEDGE(IS)                                                     READ0550
      IF (IFEDGE(IS).EQ.-1) GO TO 105                                   READ0551
      IF (IN2.EQ.IN) GO TO 70                                           READ0552
      IN4=IN+1                                                          READ0553
      DO 68 J=IN4,IN2                                                   READ0554
      NEDGE(1,J)=0                                                      READ0555
   68 NEDGE(2,J)=0                                                      READ0556
   70 CONTINUE                                                          READ0557
      IF (IFEDGE(IS).NE.0) GO TO 105                                    READ0558
      IN3=IEDGE(21)                                                     READ0559
      READ (IO5,906) JR,(IDUM(I),I=1,10)                                READ0560
      IF (JR.EQ.0) GO TO 105                                            READ0561
      N=0                                                               READ0562
      DO 75 J=1,9,2                                                     READ0563
      IF (IDUM(J).GT.0.OR.IDUM(J+1).GT.0) N=N+1                         READ0564
   75 CONTINUE                                                          READ0565
      IF (IN+N.LE.IN2) GO TO 95                                         READ0566
      N=N-(IN2-IN)                                                      READ0567
      IF (IN3.EQ.IN2) GO TO 85                                          READ0568
      IN4=IN2+1                                                         READ0569
      DO 80 J=IN4,IN3                                                   READ0570
      K=IN3-J+IN4                                                       READ0571
      NEDGE(1,K+N)=NEDGE(1,K)                                           READ0572
      NEDGE(2,K+N)=NEDGE(2,K)                                           READ0573
   80 CONTINUE                                                          READ0574
   85 CONTINUE                                                          READ0575
      IN3=IN3+N                                                         READ0576
```

```
      IN2=IN2+N
      DO 90 J=IS,21
   90 IEDGE(J)=IEDGE(J)+N
   95 CONTINUE
      DO 100 J=1,9,2
      IF (IDUM(J).EQ.0.AND.IDUM(J+1).EQ.0) GO TO 100
      IN=IN+1
      NEDGE(1,IN)=100*IDUM(J)+JR
      NEDGE(2,IN)=IDUM(J+1)
  100 CONTINUE
      GO TO 70
  105 CONTINUE
      CALL BUGOUT(8HIEDGE   ,IEDGE,1,21,21,1,1,1,1)
      CALL BUGOUT(8HNEDGE   ,NEDGE,1,2,2,1,IEDGE(21),1000,1)
C
      IF (BP(IS).EQ.0.0) GO TO 110
      READ (IO5,904) (BPFRAC(I,IS),I=1,NR)
  110 CONTINUE
C
C      BOC REG BURNUP
C
      IF (IBOC.EQ.-1) GO TO 120
      DO 115 I=1,15
  115 BOCBU(I,IS)=0.0
      IF (IBOC.NE.1) GO TO 120
      READ (IO5,904) (BOCBU(I,IS),I=1,NR)
  120 CONTINUE
C
C      END OF CYCLE DATA
C
      GO TO 40
  200 CONTINUE
      WRITE (IO6,902)
      CALL EXIT
  150 CONTINUE
      RETURN
```

```
C
  901 FORMAT(20A4)
  902 FORMAT(1H1,9X,26HLAST CASE COMPLETED - EXIT      )
  903 FORMAT(8I3,8E6.0)
  904 FORMAT(12F6.2)
  905 FORMAT(6E12.5)
  906 FORMAT(12I6)
  907 FORMAT(6E12.5)
  909 FORMAT(2I3,A3,I3,2F6.0)
  910 FORMAT(24I3)
C
      END
```

```
      SUBROUTINE PREP
C
C     SUBROUTINE PREP  GENERATES DEFAULT OR AUROMATED DATA AND SETUP
C     FOR PROBLEM EXECUTION
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
      COMMON /EDGE/ IDGE(15,20)
C
      DIMENSION DBIAS(20)
      DIMENSION NCDATA(4,10)
      DATA NCDATA / 121, 12,32,0, 157,15,36,0, 193,15,44,0, 217,17,44,0,
     1             24*0 /
      DATA DHGHT,DBIAS / 12.0, 1.00567, 1.007425, 1.00902, 1.0099,
     1                   16*1.01023  /
C
      IERR=0
C     GENERATE AXIAL LEAKAGE BIAS FROM DEFAULT
```

```
      IF (HEIGHT.EQ.0.0) GO TO 10                                       PREP0037
      FACTOR=(DHGHT+0.5)/(HEIGHT+0.5)                                   PREP0038
      DO 5 I=1,NBU                                                      PREP0039
    5 BIAS(I)=FACTOR*(DBIAS(I)-1.0) + 1.0                               PREP0040
   10 CONTINUE                                                          PREP0041
C     SET UP CYCLE 1                                                    PREP0042
      NR=NREG(1)                                                        PREP0043
      DO 15 I=1,NR                                                      PREP0044
      NAREG(I,1)=NAC1(I)                                                PREP0045
      JSHUF(I,1)=I                                                      PREP0046
      IDXREG(I,1)=100+I                                                 PREP0047
   15 CONTINUE                                                          PREP0048
      IC1=1                                                             PREP0049
      NC1=NAREG(I,1)                                                    PREP0050
      IF (NC1.EQ.1) IC1=2                                               PREP0051
      IF (NC1.EQ.1) NC1=NC1+NAREG(I,2)                                  PREP0052
C     GENERATE SHUFFLE ARRAY                                            PREP0053
      JC1=0                                                             PREP0054
      DO 60 I=2,20                                                      PREP0055
      IF (ISURCH(I).EQ.0) GO TO 60                                      PREP0056
      IF (ISHUF(I).LE.0) GO TO 49                                       PREP0057
      NCA=0                                                             PREP0058
      ICA=MOD(ISHUF(I)+1,2)                                             PREP0059
      IF (JC1.GE.NC1) ICA=0                                             PREP0060
      NF=NFEED(I)                                                       PREP0061
      IF (NF.EQ.0) GO TO 21                                             PREP0062
      DO 20 J=1,NF                                                      PREP0063
   20 NCA=NCA+NAFEED(I,J)                                               PREP0064
   21 CONTINUE                                                          PREP0065
      NR=NF                                                             PREP0066
      NR1=0                                                             PREP0067
      IF (NCA+ICA.GE.NASS) GO TO 30                                     PREP0068
      NRR=NREG(I-1)                                                     PREP0069
      DO 25 J=1,NRR                                                     PREP0070
      JJ=NRR-J+1                                                        PREP0071
      NCA=NCA+NAREG(JJ,I-1)                                             PREP0072
```

```
      NR1=NR1+1
      NR=NR+1
      IF (NCA+ICA.GE.NASS) GO TO 30
   25 CONTINUE
   30 CONTINUE
      NREG(I)=NR+ICA
      NCA=0
      NR=NR+ICA
      IF (NF.EQ.0) GO TO 36
      DO 35 J=1,NF
      JJ=NR-J+1
      NAREG(JJ,I)=NAFEED(I,NF-J+1)
      NCA=NCA+NAFEED(I,NF-J+1)
      JSHUF(JJ,I)=NF-J+1
      IDXREG(JJ,I)=100*I+NF-J+1
   35 CONTINUE
   36 CONTINUE
      IF (NR1.EQ.0) GO TO 41
      DO 40 J=1,NR1
      J1=NR-NF-J+1
      J2=NRR-J+1
      JSHUF(J1,I)=100*(I-1)+J2
      IDXREG(J1,I)=IDXREG(J2,I-1)
      NAREG(J1,I)=MINO(NAREG(J2,I-1),NASS-NCA-ICA)
      NCA=NCA+NAREG(J1,I)
   40 CONTINUE
   41 CONTINUE
      IF (ICA.EQ.0) GO TO 45
      NAREG(1,I)=1
      JC1=JC1+1
      J=1
      IF (JC1.GT.NAREG(1,1)) J=2
      JSHUF(1,I)=100+J
      IDXREG(1,I)=100+J
   45 CONTINUE
      GO TO 56
```

```
   49 CONTINUE                                                          PREP0109
      NR=NREG(I)                                                        PREP0110
      DO 55 J=1,NR                                                      PREP0111
      JK=JSHUF(J,I)                                                     PREP0112
      JJ=JK/100                                                         PREP0113
      K=MOD(JK,100)                                                     PREP0114
      IF (JJ.EQ.0) GO TO 50                                             PREP0115
      IDXREG(J,I)=IDXREG(JJ,K)                                          PREP0116
      GO TO 55                                                          PREP0117
   50 IDXREG(J,I)=100*I+K                                               PREP0118
   55 CONTINUE                                                          PREP0119
   56 CONTINUE                                                          PREP0120
      CALL BUGOUT(8HJSHUF   ,JSHUF,1,NR,15,I,I,21,1)                    PREP0121
      CALL BUGOUT(8HIDXREG  ,IDXREG,1,NR,15, I, I,21,1)                 PREP0122
   60 CONTINUE                                                          PREP0123
C     SHUFFLE FOR EQUILIBRIUM CYCLE                                     PREP0124
      IF (ISHUF(21).LE.0.OR.ISURCH(21).EQ.0) GO TO 90                   PREP0125
      NF=NFEED(21)                                                      PREP0126
      NA=0                                                              PREP0127
      NR=0                                                              PREP0128
   62 CONTINUE                                                          PREP0129
      DO 65 I=1,NF                                                      PREP0130
      NR=NR+1                                                           PREP0131
      NA=NA+NAFEED(21,I)                                                PREP0132
      IF (NA.GE.NASS) GO TO 70                                          PREP0133
   65 CONTINUE                                                          PREP0134
      IF (NR.LT.15) GO TO 62                                            PREP0135
      CALL EXIT                                                         PREP0136
   70 CONTINUE                                                          PREP0137
      NREG(21)=NR                                                       PREP0138
      NA=0                                                              PREP0139
   72 CONTINUE                                                          PREP0140
      DO 75 I=1,NF                                                      PREP0141
      NAREG(NR,21)=MIN0(NAFEED(21,I),NASS-NA)                           PREP0142
      NA=NA+NAREG(NR,21)                                                PREP0143
      IDXREG(NR,21) =2100+I                                             PREP0144
```

```
      IF  (NR+I-1.EQ.NREG(21))JSHUF(NR,21)=I
      IF  (NR+I-1.LT.NREG(21))JSHUF(NR,21)=2100+NR+NF
      NR=NR-1
      IF  (NA.GE.NASS)  GO TO 80
   75 CONTINUE
      GO TO 72
   80 CONTINUE
      NR=NREG(21)
      CALL BUGOUT(8HJSHUF    ,JSHUF,1,NR,15,21,21,21,1)
      CALL BUGOUT(8HIDXREG   ,IDXREG,1,NR,15,21,21,21,1)
   90 CONTINUE
      IF  (ISHUF(21).NE.0.OR.ISURCH(21).EQ.0)  GO TO 100
      NF=NFEED(21)
      NR=NREG(21)
      DO 98 JR=1,NR
      JK=JSHUF(JR,21)
   92 CONTINUE
      IF  (JK.LT.100)  GO TO 93
      K=MOD(JK,100)
      IF  (K.GT.NR.OR.K.LT.1)  GO TO 98
      JK=JSHUF(K,21)
      GO TO 92
   93 CONTINUE
      IDXREG(JR,21)=2100+JK
   98 CONTINUE
      CALL BUGOUT(8HIDXREG   ,IDXREG,1,NR,15,21,21,21,1)
  100 CONTINUE
C
C     GENERATE EDGE DATA
      JJJJ=0
      DO 105 I=1,10
      IF(NASS.EQ.NCDATA(1,I).AND.NADIAM.EQ.NCDATA(2,I))  JJJJ=I
  105 CONTINUE
      DO 500 IC=1,21
      IF  (ISURCH(IC).EQ.0)  GO TO 500
      IF  (IFEDGE(IC).LE.0)  GO TO 500
```

```
      IF (JJJJ.GT.0) GO TO 110
      WRITE (I06,901) NASS,NADIAM
      CALL EXIT
  110 CONTINUE
      DO 95 I=1,15
      DO 95 J=1,20
   95 IDGE(I,J)=0
C     NUMBER OF EXTERIOR EDGES = N1
      N1=4*NADIAM
C     NUMBER OF ASSEMLLIES ON PERIPHERY = N2
      N2=NCDATA(3,JJJJ)
C     NUMBER OF ASSEMBLIES AT INSIDE CORNER OF PERIGHERY = N3
      N3=N1-N2-4
C     NUMBER OF PERIPHERAL ASSEMBLIES WITH 2 EDGES ON EXTERIOR = N4
      N4=N1-N2
C
C     IDGE( ,17) - NO. ASSEBLIES WITH 2 EDGES ON PERIPHERY
C     IDGE( ,18) - NO. ASSEBLIES WITH 1 EDGES ON PERIPHERY
C     IDGE( ,19) - NO. ASSEMBLIES ON INSIDE CORNER OF PERIPHERY
C     IDGE( ,20) - NO. OF EDGES USED SO FAR
C
C     DETERMINE PERIPHERAL EDGES
      NR=NREG(IC)
      IR=NR
      NA=NAREG(NR,IC)
      NN=N4
C     2 EDGES ON PERIPHERY
  115 CONTINUE
      NUSE=MINO(NA,NN)
      IDGE(IR,20)=IDGE(IR,20)+2*NUSE
      IDGE(IR,16)=IDGE(IR,16)+2*NUSE
      IDGE(IR,17)=NUSE
      NN=NN-NUSE
      NA=NA-NUSE
      I2P=IR
      IF (NA.GT.0) GO TO 120
```

```
      IR=IR-1
      NA=NAREG(IR,IC)
      IF (NN.GT.0) GO TO 115
  120 CONTINUE
      CALL BUGOUT(8HIDGE-A  ,IDGE,1,NR,15,16,20,20,1)
C     I EDGE ON PERIPHERY
      NN=N2-N4
      J1P=IR
  125 CONTINUE
      NUSE=MINO(NA,NN)
      IDGE(IR,20)=IDGE(IR,20)+NUSE
      IDGE(IR,16)=IDGE(IR,16)+NUSE
      IDGE(IR,18)=NUSE
      I1P=IR
      NN=NN-NUSE
      NA=NA-NUSE
      IF (NA.GT.0) GO TO 130
      IR=IR-1
      NA=NAREG(IR,IC)
      IF (NN.GT.0) GO TO125
  130 CONTINUE
      CALL BUGOUT(8HIDGE-B  ,IDGE,1,NR,15,16,20,20,1)
      NA=IDGE(I1P,18)
      IR=I1P
      DO 140 I=I2P,NR
      J=NR-1+I2P
      NN=IDGE(J,17)
  135 NUSE=MINO(2*NN,NA)
      IDGE(IR,20)=IDGE(IR,20)+NUSE
      IDGE(IR,J )=IDGE(IR,J )+NUSE
      IDGE(J ,20)=IDGE(J ,20)+NUSE
      IDGE(J ,IR)=IDGE(J ,IR)+NUSE
      NN=NN-NUSE/2
      NA=NA-NUSE
      IF (NA.GT.0) GO TO 140
      IR=IR-1
```

```
      NA=IDGE(IR,18)                                                    PREP0253
      IF (NA.EQ.0) GO TO 140                                            PREP0254
      IF (NN.GT.0) GO TO 135                                            PREP0255
  140 CONTINUE                                                          PREP0256
      CALL BUGOUT(8HIDGE-C  ,IDGE,1,NR,15,1,NR,20,1)                    PREP0257
      CALL BUGOUT(8HIDGE-C  ,IDGE,1,NR,15,16,20,20,1)                   PREP0258
      IF (J1P.GT.I1P) GO TO 150                                         PREP0259
      NIN=IDGE(I1P,18)                                                  PREP0260
      NUSE=3*NIN-IDGE(I1P,20)+4*IDGE(I1P,17)-2*N3                       PREP0261
      IF (NUSE.LE.0) GO TO 180                                          PREP0262
      IDGE(I1P,I1P)=IDGE(I1P,I1P)+NUSE                                  PREP0263
      IDGE(I1P,20 )=IDGE(I1P,20 )+NUSE                                  PREP0264
      GO TO 180                                                         PREP0265
  150 CONTINUE                                                          PREP0266
      IR=J1P                                                            PREP0267
      JR=IR-1                                                           PREP0268
  152 CONTINUE                                                          PREP0269
      NN=IDGE(IR,20)                                                    PREP0270
      NIN=IDGE(IR,18)                                                   PREP0271
      NEED=4*NIN-NIN-NN+4*IDGE(IR,17)                                   PREP0272
  157 CONTINUE                                                          PREP0273
      JN=0                                                              PREP0274
      DO 160 I=1,16                                                     PREP0275
  160 JN=JN+IDGE(JR,I)                                                  PREP0276
      JNEED=3*IDGE(JR,18)-JN+4*IDGE(JR,17)                              PREP0277
  161 NUSE=MINO(NEED,JNEED)                                             PREP0278
      IDGE(IR,JR)=IDGE(IR,JR)+NUSE                                      PREP0279
      IDGE(IR,20)=IDGE(IR,20)+NUSE                                      PREP0280
      IDGE(JR,20)=IDGE(JR,20)+NUSE                                      PREP0281
      IDGE(JR,IR)=IDGE(JR,IR)+NUSE                                      PREP0282
      JNEED=JNEED-NUSE                                                  PREP0283
      NEED=NEED-NUSE                                                    PREP0284
      IF (IR.EQ.JR) GO TO 180                                           PREP0285
      IF (JNEED.GT.0) GO TO 165                                         PREP0286
      NEED=JNEED                                                        PREP0287
      IR=JR                                                             PREP0288
```

```
      JR=MAX0(JR-1,I1P)
      GO TO 157
  165 CONTINUE
      IF (NEED.EQ.0) GO TO 170
      IF (JR.EQ.I1P) GO TO 166
      JR=JR-1
      GO TO 157
  166 CONTINUE
      JR=IR
      JNEED=NEED
      GO TO 161
  170 CONTINUE
      IF (JR.EQ.I1P) GO TO 180
      IR=JR-1
      JR=MAX0(IR-1,I1P)
      GO TO 152
  180 CONTINUE
  185 CONTINUE
      CALL BUGOUT(8HIDGE-D  ,IDGE,1,NR,15,16,20,20,1)
      CALL BUGOUT(8HIDGE-D  ,IDGE,1,NR,15,1,NR,20,1)
C     ASSEMBLY ON INSIDE CORNER OF PERIPHERY
      NN=N3
      JIC=NR+1
      DO 190 I=1,NR
      J=NR-I+1
      NA=NAREG(J,IC)-IDGE(J,17)-IDGE(J,18)
      IF (NA.EQ.0) JIC=J
      IF (NA.EQ.0) GO TO 190
      NUSE=MIN0(NN,NA)
      IDGE(J,19)=NUSE
      IIC=J
      NN=NN-NUSE
      IF (NN.EQ.0) GO TO 195
  190 CONTINUE
  195 CONTINUE
      JIC=JIC-1
```

```
      DO   210 I=IIC,JIC
      J=JIC-I +IIC
      NN=IDGE(J,19)
      IF (NN.EQ.0) GO TO 210
      JR=NR
  200 NA=4*IDGE(JR,17)+3*IDGE(JR,18)+2*IDGE(JR,19)-IDGE(JR,20)
      IF (NA.LE.1) GO TO 205
      IF (JR.EQ.J) NA=NA/2
      NUSE=MIN0(NA/2,NN)
      NUSE=NUSE*2
      IDGE(JR,J )=IDGE(JR,J )+NUSE
      IDGE(JR,20)=IDGE(JR,20)+NUSE
      IDGE(J ,20)=IDGE(J ,20)+NUSE
      IDGE(J ,JR)=IDGE(J ,JR)+NUSE
      NA=NA-NUSE
      NN=NN-NUSE/2
      IF (NA.GT.1) GO TO 210
  205 JR=JR-1
      IF (IDGE(JR,20).EQ.0) GO TO 210
      GO TO 200
  210 CONTINUE
      CALL BUGOUT(8HIDGE-E  ,IDGE,1,NR,15,1,NR,20,1)
      CALL BUGOUT(8HIDGE-E  ,IDGE,1,NR,15,16,20,20,1)
C     ASSEMBLIES ON INTERIOR IN RINGS
      IF (IFEDGE(IC).NE.1) GO TO 300
      IRING=1
      NRING=1
      KRING=0
      IC1=MOD(NASS,2)
      NEI=4*(2*(1-IC1)+IC1)
      IF (IC1.EQ.0) NRING=4
      NK=NRING
      DO 260 JR=1,NR
      NA=NAREG(JR,IC)-IDGE(JR,17)-IDGE(JR,18)-IDGE(JR,19)
  215 CONTINUE
      IF (NA.EQ.0) GO TO 260
```

```
      NUSE=MINO(NRING,NA)
      NRING=NRING-NUSE
      NA=NA-NUSE
      IF (KRING.GT.0) GO TO 220
      NE=MINO(NEI,4*NUSE)
      IDGE(JR,JR)=IDGE(JR,JR)+4*NUSE-NE
      IDGE(JR,20)=IDGE(JR,20)+4*NUSE-NE
      KRING=IRING
      GO TO 250
  220 CONTINUE
      IF (KRING.EQ.IRING) GO TO 230
C     PREVIOUS TIME FILLED LAST RING - INCREASE EDGES BY 8
      NEK=4*(2*(KRING-IC1)+IC1)
      NEI=4*(2*(IRING-IC1)+IC1)
      NE1=MINO(NUSE,NEK)
      NE3=MINO(NEI,NUSE+8)
      NE2=4*NUSE-NE1-NE3
      IDGE(JR,JR)=IDGE(JR,JR)+NE2
      IDGE(JR,20)=IDGE(JR,20)+NE2+NE1
      DO 225 J=1,JR
      NE=4*NAREG(J,IC)-IDGE(J,20)
      NE=MINO(NE,NE1)
      IDGE(JR,J )=IDGE(JR,J )+NE
      IDGE(J ,JR)=IDGE(J ,JR)+NE
      IDGE(J ,20)=IDGE(J ,20)+NE
      NE1=NE1-NE
      NEK=NEK-NE
  225 CONTINUE
      GO TO 250
  230 CONTINUE
C     FILL IN SAME RING AS BEFORE
      NE1=MINO(NUSE,NEK)
      NE3=NE1
      NE2=4*NUSE-NE1-NE3
      NEK=NEK-NE1
      IDGE(JR,JR)=IDGE(JR,JR)+NE2
```

```
      IDGE(JR,20)=IDGE(JR,20)+NE2   +NE1                                PREP0397
      DO 235 J=1,JR                                                     PREP0398
      NE=4*NAREG(J,IC)-IDGE(J,20)                                       PREP0399
      NE=MIN0(NE,NE1)                                                   PREP0400
      IDGE(JR,J )=IDGE(JR,J )+NE                                        PREP0401
      IDGE(J ,JR)=IDGE(J ,JR)+NE                                        PREP0402
      IDGE(J ,20)=IDGE(J ,20)+NE                                        PREP0403
      NE1=NE1-NE                                                        PREP0404
      NEK=NEK-NE                                                        PREP0405
  235 CONTINUE                                                          PREP0406
  250 CONTINUE                                                          PREP0407
      CALL BUGOUT(8HIDGE-F  ,IDGE,1,NR,15,1,NR,20,1)                    PREP0408
      IF (NRING.GT.0) GO TO 260                                         PREP0409
      KRING=IRING                                                       PREP0410
      IRING=IRING+1                                                     PREP0411
      NRING=2*(IRING-1)*(IRING+2)+1                                     PREP0412
      IF (IC1.EQ.1) GO TO 255                                           PREP0413
      ARING=3.14*IRING**2+3.0                                           PREP0414
      NRING=0.125*ARING                                                 PREP0415
      NRING=8*NRING                                                     PREP0416
  255 CONTINUE                                                          PREP0417
      NRING=NRING-NK                                                    PREP0418
      NK=NK+NRING                                                       PREP0419
      IF (NA.GT.0) GO TO 215                                            PREP0420
  260 CONTINUE                                                          PREP0421
  265 CONTINUE                                                          PREP0422
C     COMBINE INTERIOR TO PERIPHERY                                     PREP0423
      JR=1                                                              PREP0424
      NA1=4*NAREG(JR,IC)-IDGE(JR,20)                                    PREP0425
      DO 280 J=1,NR                                                     PREP0426
      JJ=NR-J+1                                                         PREP0427
      NA2=4*NAREG(JJ,IC)-IDGE(JJ,20)                                    PREP0428
      IF (NA2.EQ.0) GO TO 280                                           PREP0429
  270 IF (NA1.GT.0) GO TO 275                                           PREP0430
      JR=JR+1                                                           PREP0431
      IF (JR.GT.NR) GO TO 280                                           PREP0432
```

```
      NA1=4* NAREG(JR,IC)-IDGE(JR,20)
      GO TO 270
  275 CONTINUE
      NUSE=MINO(NA1,NA2)
      NA1=NA1-NUSE
      NA2=NA2-NUSE
      IF (JJ.EQ.JR) NUSE=NUSE/2
      IDGE(JJ,JR)=IDGE(JJ,JR)+NUSE
      IDGE(JJ,20)=IDGE(JJ,20)+NUSE
      IDGE(JR,20)=IDGE(JR,20)+NUSE
      IDGE(JR,JJ)=IDGE(JR,JJ)+NUSE
      IF (NA2.EQ.0) GO TO 280
      IF (NA1.EQ.0) GO TO 270
  280 CONTINUE
      CALL BUGOUT(8HIDGE-G  ,IDGE,1,NR,15,1,NR,20,1)
      CALL BUGOUT(8HIDGE-G  ,IDGE,1,NR,15,16,20,20,1)
      GO TO 450
  300 CONTINUE
C
C     CHECKERBOARD PATTER
C
      NRM=1
      IF (NAREG(1,IC).NE.1) GO TO 305
      NRM=2
      IDGE(3,1)=4
      IDGE(1,3)=4
      IDGE(1,20)=4
      IDGE(3,20)=IDGE(3,20)+4
  305 CONTINUE
C      NP= NO. EDGES LEFT ON PERIPHERAL ASSEMBLIES
      NP=0
      NR1=NR
      DO 310 JR=1,NR
      IR=NR-JR+1
      K=IDGE(IR,17)+IDGE(IR,18)+IDGE(IR,19)
      IF (K.EQ.0) GO TO 315
```

```
      NR1=IR                                                            PREP0469
      NP=NP+4*K-IDGE(IR,20)                                             PREP0470
  310 CONTINUE                                                          PREP0471
  315 CONTINUE                                                          PREP0472
      IF (K.EQ.NAREG(NR1,IC)) NR1=NR1-1                                 PREP0473
      NR2=MAX0(NRM,NR1-1)                                               PREP0474
C     NR1- NR2  REGIONS IN INTERIOR TO FACE PERIPHERY ASSEMBLIES        PREP0475
  320 NSUM=0                                                            PREP0476
      DO 325 K=NR2,NR1                                                  PREP0477
  325 NSUM=NSUM+NAREG(K,IC)-IDGE(K,17)-IDGE(K,18)-IDGE(K,19)            PREP0478
      IF (NSUM.GE.NP/2) GO TO 330                                       PREP0479
      IF (NR2.EQ.1) GO TO 330                                           PREP0480
      NR2=NR2-1                                                         PREP0481
      GO TO 320                                                         PREP0482
  330 CONTINUE                                                          PREP0483
      DO 350 IR=NR1,NR                                                  PREP0484
      K=IDGE(IR,17)+IDGE(IR,18)+IDGE(IR,19)                             PREP0485
      IF (K.EQ.0) GO TO 350                                             PREP0486
      NE=4*K-IDGE(IR,20)                                                PREP0487
      A=NE                                                              PREP0488
      DO 340 JR=NR2,NR1                                                 PREP0489
      B=NAREG(JR,IC)-IDGE(JR,17)-IDGE(JR,18)-IDGE(JR,19)                PREP0490
      B=B/NSUM                                                          PREP0491
      NUSE=A*B+0.5                                                      PREP0492
      N1=4*NAREG(JR,IC)-IDGE(JR,20)                                     PREP0493
      IF (JR.EQ.NR1) NUSE=NE                                            PREP0494
      NUSE=MIN0(NE,NUSE)                                                PREP0495
      IF (IR.EQ.JR) N1=N1/2                                             PREP0496
      NUSE=MIN0(NUSE,N1)                                                PREP0497
      NE=NE-NUSE                                                        PREP0498
      IDGE(IR,JR)=IDGE(IR,JR)+NUSE                                      PREP0499
      IDGE(JR,IR)=IDGE(JR,IR)+NUSE                                      PREP0500
      IDGE(JR,20)=IDGE(JR,20)+NUSE                                      PREP0501
      IDGE(IR,20)=IDGE(IR,20)+NUSE                                      PREP0502
  340 CONTINUE                                                          PREP0503
  350 CONTINUE                                                          PREP0504
```

```
      CALL BUGOUT(8HIDGE-H  ,IDGE,1,NR,15,1,NR,20,1)
      CALL BUGOUT(8HIDGE-H  ,IDGE,1,NR,15,16,20,20,1)
      NSUM=0
      DO 355 IR=NRM,NR
      K=4*NAREG(IR,IC)-IDGE(IR,20)
      IF (K.EQ.0) GO TO 355
      NSUM=NSUM+K
      NR2=IR
  355 CONTINUE
      NR1=NR2-1
      DO 380 IR=NRM,NR1
      NE=4*NAREG(IR,IC)-IDGE(IR,20)
      A=NE
      IF (NSUM-NE.LE.0) GO TO 380
      A=A/(NSUM-NE)
      KR=IR+1
      DO 370 JR=KR,NR2
      K=4*NAREG(JR,IC)-IDGE(JR,20)
      NUSE=A*K+0.5
      NUSE=MINO(NUSE,NE,K)
      NE=NE-NUSE
      NSUM=NSUM-2*NE
      IDGE(IR,JR)=IDGE(IR,JR)+NUSE
      IDGE(IR,20)=IDGE(IR,20)+NUSE
      IDGE(JR,20)=IDGE(JR,20)+NUSE
      IDGE(JR,IR)=IDGE(JR,IR)+NUSE
  370 CONTINUE
  380 CONTINUE
      CALL BUGOUT(8HIDGE-I  ,IDGE,1,NR,15,1,NR,20,1)
      CALL BUGOUT(8HIDGE-I  ,IDGE,1,NR,15,16,20,20,1)
      DO 390 IR=1,NR
      K=4*NAREG(IR,IC)-IDGE(IR,20)
      IDGE(IR,IR)=IDGE(IR,IR)+K
  390 CONTINUE
      CALL BUGOUT(8HIDGE-J  ,IDGE,1,NR,15,1,NR,20,1)
      CALL BUGOUT(8HIDGE-J  ,IDGE,1,NR,15,16,20,20,1)
```

```
C
C     PACK IDGE INTO NEDGE ARRAY
  450 CONTINUE
      N=0
      CALL BUGOUT(8HCYCLE    ,IC,1,1,1,1,1,1,1)
      CALL BUGOUT(8HIDGE     ,IDGE,1,NR,15,1,20,20,1)
      DO 460 J=1,NR
      DO 460 JJ=J,16
      IF (IDGE(J,JJ).GT.0) N=N+1
  460 CONTINUE
      IN=0
      IF (IC.GT.1) IN=IEDGE(IC-1)
      IN2=IEDGE(IC)
      IN3=IEDGE(21)
      IF (IN2-IN.GE.N) GO TO 480
      N=N-(IN2-IN)
      IF (IN2.EQ.IN3) GO TO 470
      IN4=IN2+1
      DO 465 J=IN4,IN3
      K=IN3-J+IN4
      NEDGE(1,K+N)=NEDGE(1,K)
      NEDGE(2,K+N)=NEDGE(2,K)
  465 CONTINUE
  470 CONTINUE
      DO 475 J=IC,21
  475 IEDGE(J)=IEDGE(J)+N
  480 CONTINUE
      N=0
      DO 490 J=1,NR
      DO 490 JJ=J,16
      IF (IDGE(J,JJ).EQ.0) GO TO 490
      IN=IN+1
      K=JJ
      IF (JJ.EQ.16) K=0
      NEDGE(1,IN)=100*K+J
      NEDGE(2,IN)=IDGE(J,JJ)
```

```
  490 CONTINUE
  500 CONTINUE
C     SET UP NUMBER OF ASM DISCHARGED EACH CYCLE
      DO 510 IC=1,21
      DO 510 IR=1,15
  510 NADIS(IR,IC)=0
      DO 530 IC=1,20
      IF (ISURCH(IC).EQ.0) GO TO 530
      NR=NREG(IC)
      IC1=IC+1
      DO 525 IR=1,NR
      NADIS(IR,IC)=NAREG(IR,IC)
      IF (IC.EQ.20) GO TO 525
      K=100*IC+IR
      DO 520 JC=IC1,20
      IF (ISURCH(JC).EQ.0) GO TO 520
      NR1=NREG(JC)
      DO 515 JR=1,NR1
      IF (JSHUF(JR,JC).NE.K) GO TO 515
      NADIS(IR,IC)=NADIS(IR,IC)-NAREG(JR,JC)
  515 CONTINUE
  520 CONTINUE
      IF (NADIS(IR,IC).LT.0) IERR=1
      IF (NADIS(IR,IC).LT.0) WRITE (I06,902) IR,IC
  525 CONTINUE
      CALL BUGOUT(8HNADIS   ,NADIS,1,NR,15,IC,IC,21,1)
  530 CONTINUE
C      EQUILLIBRIUM CYCLE
      NR=NREG(21)
      IF (ISURCH(21).EQ.0) GO TO 545
      DO 540 IR=1,NR
      NADIS(IR,21)=NAREG(IR,21)
      K=2100+IR
      DO 535 JR=1,NR
      IF (JSHUF(JR,21).NE.K) GO TO 535
      NADIS(IR,21)=NADIS(IR,21)-NAREG(JR,21)
```

```
  535 CONTINUE
  540 CONTINUE
      CALL BUGOUT(8HNADIS   ,NADIS,1,NR,15,21,21,21,1)
  545 CONTINUE
C
C     GENERATE ISOTOPIC DATA FROM FIT
C
      IF (ISO.NE.-1) GO TO 570
      ISO=0
      DO 565 I=1,NRICH
      E=ENRICH(I)
      DO 560 J=1,NBU
      B=BURNUP(J)
      ISOTOP(J,I,1)=EXP(-0.00159*B*(1.0-4.1E-3*B))
      ISOTOP(J,I,2)=E*EXP(-0.1162*B*(1.0+2.75E-5*B*B*E)/E**0.965)
      ISOTOP(J,I,3)=4.795*E**0.40*(1.0-EXP(-.1425*B/E**0.6))
  560 CONTINUE
  565 CONTINUE
  570 CONTINUE
      IF (IERR.EQ.1) CALL EXIT
      RETURN
C
  901 FORMAT(1H0,9X,48H*** ERROR, CAN NOT GENERATE EDGE DATA FOR A CORE,
     1      6H WITH ,I3,16H ASSEMBLIES AND ,I2,18H ASSEMBLIES ACROSS  )
  902 FORMAT(1H0,9X,47HERROR, NEGATIVE NUMBER OF ASSEMBLIES DISCHARGED ,
     1      12H FROM REGION,I3,7H, CYCLE ,I3 /)
C
      END
```

```
      SUBROUTINE INEDIT
C
C     SUBROUTINE INEDIT EDITS OUT BASIC INPUT DATA
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
C
      DIMENSION ISONAM(3)
      DATA NCASE / 0 /
      DATA ISONAM / 4H U   , 4HU-35, 4H PU   /
C
      NCASE=NCASE+1
      CALL HEAD
      WRITE (IO6,101) NASS,NADIAM,HEIGHT,ULOAD,POWER
      WRITE (IO6,102) W,ALPHA,ELIM,BUDIF,CYBIAS
      WRITE (IO6,114) BWRTH,DPWRTH,XEWRTH
      IF (IHAL.GT.0) WRITE (IO6,103)
      IF (IFCOST.GT.0) WRITE (IO6,104)
```

```
      IF (NCASE.GT.1) GO TO 25
      DO 12 N1=1,NRICH,8
      N2=MIN0(NRICH,N1+7)
      CALL HEAD
      WRITE (I06,105)
      WRITE (I06,106)  (ENRICH(I),I=N1,N2)
      WRITE (I06,107)  (ITYPE(I),I=N1,N2)
      WRITE (I06,108)  (POWFIS(I),I=N1,N2)
      WRITE (I06,109)
      DO 10 I=1,NBU
      WRITE (I06,110) BURNUP(I),BIAS(I),BPWRTH(I),(KINF(I,J),J=N1,N2)
   10 CONTINUE
   12 CONTINUE
      NISO=3
      N1=1
      N2=NRICH
      DO 20 K=1,NISO
      CALL HEAD
      WRITE (I06,111) ISONAM(K)
      WRITE (I06,112)  (ENRICH(I),I=N1,N2)
      DO 15 I=1,NBU
      WRITE (I06,113)  BURNUP(I), (ISOTOP(I,J,K),J=N1,N2)
   15 CONTINUE
   20 CONTINUE
   25 CONTINUE
      RETURN
C
  101 FORMAT(1H0, 9X,24HNUMBER OF ASSEMBLIES....,I6 /
     1           10X,24HASSEMBLIES ACROSS CORE.. ,I6 /
     2           10X,24HCORE HEIGHT.............,F6.2,4H FT. /
     3           10X,24HAVERAGE CORE LOADING....,F6.2,4H MTM /
     4           10X,24HCORE THERMAL POWER......,F6.0,4H MW )
  102 FORMAT(1H0, 9X,24HPROB. OF ABSORP.  (W).....,F6.3 /
     1           10X,24HALBEDO..................,F6.3 /
     2           10X,24HMINIMUM ENRICHMENT......,F6.3, 4H W/O /
     3           10X,24HMAXIMUM ENRICHMENT......,F6.3, 4H W/O /
```

```
    4          10X,24HBURNUP VARIATION PARAM..,F6.3,8H GWD/MTM /          INED0073
    5          10X,24HCYCLE BIAS FACTOR.......,F6.3)                      INED0074
  103 FORMAT(1H0,9X,33HA HALING CALCULATION WILL BE USED  )               INED0075
  104 FORMAT(1H0,9X,32HFUEL COST DATA WILL BE GENERATED  )                INED0076
  105 FORMAT(38X,34HK EFF VERSUS BURNUP AND ENRICHMENT  //)               INED0077
  106 FORMAT(10X,10HENRICHMENT,14X,8F10.3)                                INED0078
  107 FORMAT(10X,10HFUEL TYPE ,11X,8I10)                                  INED0079
  108 FORMAT(10X,10HPOWER/FIS.,14X,8F10.3)                                INED0080
  109 FORMAT(1H0,9X,6HBURNUP,3X,4HBIAS,4X,7HBP WRTH,15X,5HK EFF /)        INED0081
  110 FORMAT(8X,F8.3,F7.4,F11.2,8F10.5)                                   INED0082
  111 FORMAT(30X,15HISOTOPIC DATA (,A4,30H) VERSUS BURNUP AND ENRICHMENT  INED0083
    1      //)                                                            INED0084
  112 FORMAT(18X,15F8.3)                                                  INED0085
  113 FORMAT(10X,F8.3,15F8.4)                                             INED0086
  114  FORMAT(1H0, 9X,24HBORON WORTH.............,F6.1,8H PCM/PPM /       INED0087
    2           10X,24HDOPPLER DEFECT...........,F6.0,4H PCM /            INED0088
    2           10X,24HXENON DEFECT.............,F6.0,4H PCM )            INED0089
C                                                                         INED0090
      END                                                                 INED0091
```

```
      SUBROUTINE CALC
C
C     CALCULATE CYCLE BY CYCLE DATA
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
      COMMON /EDGE/ RFRAC(15,20)
      COMMON /CYDEP/ BSHARE(15,10),BBU(10),BKEFF(10),NBBU
C
      DIMENSION REGE(15),IRTYPE(15),REGBU(15),RLOAD(15)
C
      DO 100 IC=1,20
      IF (ISURCH(IC).EQ.0) GO TO 100
      IN1=1
      IF (IC.GT.1) IN1=IEDGE(IC-1)+1
      IN2=IEDGE(IC)
      DO 5 I=1,16
      DO 5 J=1,15
```

```
    5 RFRAC(J,I)=0.0
      DO 15 I=IN1,IN2
      K=NEDGE(1,I)
      A=NEDGE(2,I)*0.25
      JR=K/100
      IR=MOD(K,100)
      IF (K.EQ.0) GO TO 15
      IF (JR.EQ.0) GO TO 10
      RFRAC(JR,IR)=A/NAREG(JR,IC)
      RFRAC(IR,JR)=A/NAREG(IR,IC)
      GO TO 15
   10 CONTINUE
      RFRAC(IR,16)=A/NAREG(IR,IC)
   15 CONTINUE
      NR=NREG(IC)
      SUM=0.0
      DO 30 IR=1,NR
      REGBU(IR)=BOCBU(IR,IC)
      K=IDXREG(IR,IC)
      JR=MOD(K,100)
      JC=K/100
      IF (JC.EQ.1) GO TO 20
      REGE(IR)=FRICH(JC,JR)
      IRTYPE(IR)=FTYPE(JC,JR)
      RLOAD(IR)=FLOAD(JC,JR)
      SUM=SUM+RLOAD(IR)*NAREG(IR,IC)
      GO TO 30
   20 CONTINUE
      REGE(IR)=C1RICH(JR)
      IRTYPE(IR)=C1TYPE(JR)
      RLOAD(IR)=C1LOAD(JR)
      SUM=SUM+RLOAD(IR)*NAREG(IR,IC)
   30 CONTINUE
      CYLOAD(IC)=SUM
      IF (CYBU(IC).EQ.0.0) CYBU(IC)=1.E-5*CAPFAC(IC)*CYTIM(IC)*POWER/
     1      CYLOAD(IC)
```

```
      SUM=ANASS/SUM
      DO 35 IR=1,NR
   35 RLOAD(IR)=RLOAD(IR)*SUM
      IS=ISURCH(IC)
      IF (IS.LE.10) GO TO 45
      K=100*IC+IS-10
      DO 40 IR=1,NR
      IF (IDXREG(IR,IC).EQ.K) IS=10+IR
   40 CONTINUE
   45 CONTINUE
      CALL CYCALC(IS,CYBU(IC),EOLK(IC),BP(IC),NR,REGBU,REGE,IRTYPE,
     1            NAREG(1,IC),RFRAC,BPFRAC(1,IC),RLOAD)
      IF (IS.LE.10) GO TO 50
      JS=ISURCH(IC)-10
      FRICH(IC,JS)=REGE(IS-10)
   50 CONTINUE
      DO 75 IR=1,NR
      EOCBU(IR,IC)=REGBU(IR)
      FRAC=NADIS(IR,IC)
      FRAC=FRAC/NAREG(IR,IC)
      K=100*IC+IR
      IF (IC.EQ.20) GO TO 75
      IC1=IC+1
      DO 60 JC=IC1,20
      NR1=NREG(JC)
      IF (ISURCH(JC).EQ.0) GO TO 60
      DO 55 JR=1,NR1
      IF (JSHUF (JR,JC).NE.K) GO TO 55
      IF (BOCBU(JR,JC).GT.0) GO TO 55
      BOCBU(JR,JC)=EOCBU(IR,IC)-BUDIF*FRAC
   55 CONTINUE
   60 CONTINUE
   75 CONTINUE
      CALL CYEDIT(IC,REGE,IRTYPE)
  100 CONTINUE
C
```

```
      RETURN                                                    CALC0109
      END                                                       CALC0110
```

```
      SUBROUTINE EQCALC
C
C     CALCULATE EQUILIBRIUM CYCLE
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
      COMMON /EDGE/ RFRAC(15,20)
C
      DIMENSION REGE(15),RTYPE(15),REGBU(15),PSHAR(15),RLOAD(15)
      DIMENSION INDEX(15)
      INTEGER RTYPE
C
      IS=ISURCH(21)
      IF (IS.EQ.0) RETURN
      NR=NREG(21)
      NF=NFEED(21)
      ITER=0
      DO 5 I=1,NR
```

```
    5 PSHAR(I)=1.0
      IN1=IEDGE(20)+1
      IN2=IEDGE(21)
      DO 205 I=1,16
      DO 205 J=1,15
  205 RFRAC(J,I)=0.0
   10 CONTINUE
      ITER=ITER+1
      IF (ITER.GT.50) GO TO 400
      JITER=0
   15 CONTINUE
      JITER=JITER+1
      IF (JITER.GT.50) GO TO 400
   20 CONTINUE
      SUM=0.0
      DO 25 JR=1,NR
   25 INDEX(JR)=0
      NN=0
   30 CONTINUE
      DO 50 JR=1,NR
      IF (INDEX(JR).GT.0) GO TO 50
      IF (JSHUF(JR,21).GE.100) GO TO 35
      JF=JSHUF(JR,21)
      NN=NN+1
      INDEX(JR)=1
      RLOAD(JR)=FLOAD(21,JF)
      REGE(JR)=FRICH(21,JF)
      SUM=SUM+RLOAD(JR)*NAREG(JR,21)
      RTYPE(JR)=FTYPE(21,JF)
      REGBU(JR)=0.0
      EOCBU(JR,21)=0.0
      EOCBU(JR,21)=PSHAR(JR)*CYBU(21)
      GO TO 50
   35 CONTINUE
      JK=JSHUF(JR,21)
      N=MOD(JK,100)
```

```
      IF (INDEX(N).EQ.0) GO TO 50
      KR=JR
      NN=NN+1
      INDEX(JR)=1
      FRAC=NADIS(N,21)
      FRAC=FRAC/NAREG(N,21)
      REGBU(KR)=CYBU(21)*PSHAR(N)+REGBU(N)-BUDIF*FRAC
      BOCBU(KR,21)=REGBU(KR)
      RTYPE(KR)=RTYPE(N)
      BOCBU(KR,21)=REGBU(KR)+CYBU(21)*PSHAR(KR)
      REGE(KR)=REGE(N)
      RLOAD(KR)=RLOAD(N)
      SUM=SUM+RLOAD(KR)*NAREG(KR,21)
   50 CONTINUE
      IF (NN.LT.NR) GO TO 30
      CYLOAD(21)=SUM
      SUM=ANASS/SUM
      DO 55 JR=1,NR
      RFRAC(JR,JR)=0.0
   55 RLOAD(JR)=SUM*RLOAD(JR)
      DO 215 I=IN1,IN2
      K=NEDGE(1,I)
      A=NEDGE(2,I)*0.25
      JR=K/100
      IR=MOD(K,100)
      IF (K.EQ.0) GO TO 215
      IF (JR.EQ.0) GO TO 210
      RFRAC(JR,IR)=A/NAREG(JR,21)
      RFRAC(IR,JR)=A/NAREG(IR,21)
      GO TO 215
  210 RFRAC(IR,16)=A/NAREG(IR,21)
  215 CONTINUE
      EIGEN=EOLK(21)
      CALL CYCALC(3,CYBU(21),EIGEN,BP(21),NR,REGBU,REGE,RTYPE,
     1            NAREG(1,21),RFRAC,BPFRAC(1,21),RLOAD)
      SUM=0.0
```

```
      DO 60 JR=1,NR
      PSHAR(JR)=REGBU(JR)-BOCBU(JR,21)
      SUM=SUM+PSHAR(JR)*NAREG(JR,21)
   60 CONTINUE
      SUM=ANASS/SUM
      DIF=0.0
      DO 65 JR=1,NR
      DIF=AMAX1(DIF,ABS(EOCBU(JR,21)-REGBU(JR)))
   65 PSHAR(JR)=SUM*PSHAR(JR)
      IF (DIF.GT.0.01) GO TO 15
      IF (IS.GT.10) GO TO 100
C
      GO TO (70,80,90), IS
   70 CONTINUE
      IF (ITER.GT.1) GO TO 75
      DIF=100.*(EOLK(21)-EIGEN)
      OLD=CYBU(21)
      OLDK=EIGEN
      CYBU(21)=CYBU(21)-SIGN(AMIN1(2.,ABS(DIF)),DIF)
      GO TO 10
   75 CONTINUE
      SLOPE=(CYBU(21)-OLD)/(EIGEN-OLDK)
      OLDK=EIGEN
      OLD=CYBU(21)
      CYBU(21)=CYBU(21)-SLOPE*(EIGEN-EOLK(21))
      IF (ABS(CYBU(21)-OLD).LT.0.001) GO TO 150
      GO TO 10
C
   80 CONTINUE
      IF (ITER.GT.1) GO TO 85
      DIF=1.E+5*(EOLK(21)-EIGEN)
      OLD=BP(21)
      OLDK=EIGEN
      BP(21)=BP(21)-SIGN(AMIN1(2000.,ABS(DIF)),DIF)
      GO TO 10
   85 CONTINUE
```

```
      SLOPE=(BP(21)-OLD)/(EIGEN-OLDK)
      OLDK=EIGEN
      OLD=BP(21)
      BP(21)=BP(21)-SLOPE*(EIGEN-EOLK(21))
      IF (ABS(BP(21)-OLD).LT.1.0) GO TO 150
      GO TO 10
C
   90 CONTINUE
      EOLK(21)=EIGEN
      GO TO 150
  100 CONTINUE
      JS=IS-10
      IF (ITER.GT.1) GO TO 105
      OLD=FRICH(21,JS)
      OLDK=EIGEN
      FRICH(21,JS)=FRICH(21,JS)+0.05
      GO TO 10
  105 CONTINUE
      SLOPE=(FRICH(21,JS)-OLD)/(EIGEN-OLDK)
      OLDK=EIGEN
      OLD=FRICH(21,JS)
      FRICH(21,JS)=FRICH(21,JS)-SLOPE*(EIGEN-EOLK(21))
      FRICH(21,JS)=AMAX1(ELIM(1),AMIN1(ELIM(2),FRICH(21,JS)))
      IF (ABS(FRICH(21,JS)-OLD).LT.0.001) GO TO 150
      GO TO 10
C
  400 CONTINUE
      WRITE (IO6,101)
C
C     END OF ITERATION
C
  150 CONTINUE
      EOLK(21)=EIGEN
      CALL CYEDIT(21,REGE,RTYPE)
      RETURN
C
```

```
  101 FORMAT(1H0,9X,23HEQ. CYCLE SEARCH FAILED  )                       EQC 0181
C                                                                       EQC 0182
      END                                                               EQC 0183
```

```
      SUBROUTINE CYCALC(IS,CYBU,EOLK,BP,NREG,REGBU,REGE,IRTYPE,NAREG,
     1                  RFRAC,BPFRAC,RLOAD)
C
C     SUBROUTINE CYCALC - INDIVIDUAL CYCLE CALCULATION WITH OPTIONAL
C           SEARCH
C
C           IS     SEARCH PARAMETER
C                  =1 CALCULATE CYCLE BURNUP
C                  =2 CLACULATE BP PENALTY
C                  =3 CALCULATE EOL K
C                  =10+J CALCULATE FEED ENRICHMENT FOR REGION J
C           CYBU   CYCLE BURNUP DESIRED OR RETURNED AS CALCULATED
C           EOLK   EOL K DESIRED OR RETURNED AS CALCULATED
C           REGBU(15)  IN AS BOL REGION BURNUPS, OUT AS EOL BURNUPS
C           REGE(15)  REGION ENRICHMENTS
C           IRTYPE(15)  REGION FUEL TYPE
C           NAREG(15)  NUMBER OF ASSEMBLIES IN EACH REGION
C           RFRAC(15,16)  FRACTION OF EDGES ADJACENT TO EACH REGION ****
C                  ****  DESTROYED IN ROUTINE *****
C           BP     NUMBER OF BURNABLE POISON RODS
C           NREG   NUMBER OF REGIONS
C           BPFRAC(15)  FRACTION OF BP RODS IN EACH REGION
C           RLOAD(15)   REGION LOADINGS
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
```

```
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
      COMMON /CYDEP/ BSHARE(15,10),BBU(10),BKEFF(10),NBBU
C
      DIMENSION REGBU(15),REGE(15),IRTYPE(15),NAREG(15),RFRAC(15 ,16),
     1          REGBU1(15),PSHARE(15)
      DIMENSION BPFRAC(15),RLOAD(15)
      REAL*8 A(15,15),B(15),EIGEN
CDC   REAL   A(15,15),B(15),EIGEN
      DATA DELBU / 4.0 /
C
      IF (IS.EQ.1.AND.CYBU.LE.0.0) CYBU=11.
      IF (IS.EQ.2.AND.BP.EQ.0.0) BP=1000.
      JS=MOD(IS,10)
      IF (IS.GT.10.AND.REGE(JS).LE.0.0) REGE(JS)=3.0
      CALL BUGOUT(8HREGE    ,REGE,1,NREG,15,1,1,1,2)
      CALL BUGOUT(8HREGBU   ,REGBU,1,NREG,15,1,1,1,2)
C     CALL BUGOUT(8HRFRAC   ,RFRAC,1,NREG,15,1,NREG,16,2)
C     CALL BUGOUT(8HRFRAC   ,RFRAC,1,NREG,15,16,16,16,2)
C     CALL BUGOUT(8HNAREG   ,NAREG,1,NREG,15,1,1,1,1)
      DO 10 I=1,NREG
      B(I)=0.0
      DO 5 J=1,NREG
      RFRAC(I,J)=RFRAC(I,J)*4.0*W
    5 CONTINUE
      RFRAC(I,I)=RFRAC(I,I) + 1.0-4.0*W*(1.0+ALPHA*RFRAC(I,16))
   10 CONTINUE
      ITER=0
C
   15 ITER=ITER+1
      IF (ITER.GT.50) GO TO 140
      DO 20 I=1,NREG
      REGBU1(I)=REGBU(I)
   20 CONTINUE
      CYB=0.0
      IF (IHAL.EQ.1) GO TO 210
```

```
      NBBU=0
      DO 50 K=1,10
      DELB=AMIN1(DELBU,CYBU-CYB)
      PPM=0.0
   25 CONTINUE
      DO 30 I=1,NREG
      AKINF=TERP(REGBU1(I),REGE(I),IRTYPE(I),KINF,0,AKB,BPK)
      AKINF=AKINF-BP*BPFRAC(I)*BPK/NAREG(I)+PPM
      DO 30 J=1,NREG
      A(I,J)=RFRAC(I,J)*AKINF/AKB
   30 CONTINUE
C
      CALL SOLVE(A,B,EIGEN,NREG)
C
      IF (BWRTH.EQ.0.0) GO TO 34
      DIF=EIGEN-1.000
      IF (ABS(DIF).LT.1.E-4) GO TO 34
      PPM=PPM-DIF
      GO TO 25
   34 EIGEN=EIGEN-PPM
      SUM=0.0
      DO 35 I=1,NREG
      B(I)=B(I)*TERP(REGBU1(I),REGE(I),IRTYPE(I),POWFIS,2,AKB,BPK)
      B(I)=B(I)/RLOAD(I)
      SUM=SUM+B(I)*NAREG(I)
   35 CONTINUE
      SUM=SUM/ANASS
      NBBU=NBBU+1
      DO 40 I=1,NREG
      PSHARE(I)=B(I)/SUM
      BSHARE(I,NBBU)=PSHARE(I)
      REGBU1(I)=REGBU1(I)+DELB*PSHARE(I)
   40 CONTINUE
      BBU(NBBU)=CYB
      BKEFF(NBBU)=EIGEN
      CYB=CYB+DELB
```

```
      IF (CYB.GE.CYBU.AND.DELB.EQ.0.0) GO TO 55
      DELB=0.0
      IF (CYB.GE.CYBU) GO TO 25
   50 CONTINUE
   55 CONTINUE
      GO TO 250
  210 CONTINUE
C
C      HALING DISTRIBUTION CALCULATION
C
      K=1
      OLDKH=0.0
      IF (ITER.GT.1) GO TO 220
      DO 215 I=1,NREG
  215 PSHARE(I)=1.0
  220 CONTINUE
      ITERH=0
  225 ITERH=ITERH+1
      IF (ITERH.GT.50) GO TO 140
      DO 230 I=1,NREG
      REGBU1(I)=REGBU (I)+PSHARE(I)*CYBU
      AKINF=TERP(REGBU1(I),REGE(I),IRTYPE(I),KINF,0,AKB,BPK)
      AKINF=AKINF-BP*BPFRAC(I)*BPK/NAREG(I)
      DO 230 J=1,NREG
      A(I,J)=RFRAC(I,J)*AKINF/AKB
  230 CONTINUE
C
      CALL SOLVE(A,B,EIGEN,NREG)
C
      SUM=0.0
      DO 235 I=1,NREG
      B(I)=B(I)*TERP(REGBU1(I),REGE(I),IRTYPE(I),POWFIS,2,AKB,BPK)
      B(I)=B(I)/RLOAD(I)
      SUM=SUM+B(I)*NAREG(I)
  235 CONTINUE
      SUM=SUM/ANASS
```

```
      DIFH=0.0
      DO 240 I=1,NREG
      PSHARE(I)=B(I)/SUM
      BSHARE(I,K)=PSHARE(I)
      DIFH=AMAX1(DIFH,ABS(REGBU(I)+PSHARE(I)*CYBU-REGBU1(I)))
  240 CONTINUE
      NBBU=1
      BBU(1)=CYBU
      BKEFF(1)=EIGEN
      IF (DIFH.GT.0.001) GO TO 225
  250 CONTINUE
      IF (IS.GT.10) GO TO 90
      GO TO (60,70,80), IS
   60 CONTINUE
C
C       SEARCH ON CYCLE BURNUP
C
      IF (ITER.GT.1) GO TO 65
      DIF=100.0*(EOLK-EIGEN)
      OLD=CYBU
      OLDK=EIGEN
      CY BU=CYBU - SIGN(AMIN1(2.    ,ABS(DIF)),DIF)
      GO TO 15
   65 CONTINUE
      SLOPE=(CYBU-OLD)/(EIGEN-OLDK)
      OLDK=EIGEN
      OLD=CYBU
      CYBU=CYBU-SLOPE*(EIGEN-EOLK)
      IF (ABS(CYBU-OLD).LT.0.001) GO TO 150
      GO TO 15
   70 CONTINUE
C
C       SEARCH ON BP PENALTY
C
      IF (ITER.GT.1) GO TO 75
      DIF=1.E+5*(EOLK-EIGEN)
```

```
      OLD=BP
      OLDK=EIGEN
      BP=BP-SIGN(AMIN1(2000.,ABS(DIF)),DIF)
      GO TO 15
   75 CONTINUE
      SLOPE=(BP-OLD)/(EIGEN-OLDK)
      OLDK=EIGEN
      OLD=BP
      BP=BP-SLOPE*(EIGEN-EOLK)
      IF (ABS(BP-OLD).LT.1.0) GO TO 150
      GO TO 15
   80 CONTINUE
C
C      CALCULATE EOLK
C
      EOLK=EIGEN
      GO TO 150
   90 CONTINUE
C
C      SEARCH ON ENTICHMENT
C
      IF (ITER.GT.1) GO TO 95
      OLD=REGE(JS)
      OLDK=EIGEN
      REGE(JS)=REGE(JS)+0.10
      GO TO 15
   95 CONTINUE
      SLOPE=(REGE(JS)-OLD)/(EIGEN-OLDK)
      OLDK=EIGEN
      OLD=REGE(JS)
      REGE(JS)=REGE(JS)-SLOPE*(EIGEN-EOLK)
      REGE(JS)=AMAX1(ELIM(1),AMIN1(ELIM(2),REGE(JS)))
      IF (ABS(REGE(JS)-OLD).LT.0.001) GO TO 150
      GO TO 15
C
C      END OF ITERATION LOOP
```

```
C                                                                               CYC 0217
  140 CONTINUE                                                                  CYC 0218
      WRITE (IO6,101) IS,CYBU,EOLK                                              CYC 0219
  150 CONTINUE                                                                  CYC 0220
      EOLK=EIGEN                                                                CYC 0221
      DO 155 I=1,NREG                                                           CYC 0222
      REGBU(I)=REGBU1(I)                                                        CYC 0223
  155 CONTINUE                                                                  CYC 0224
      RETURN                                                                    CYC 0225
C                                                                               CYC 0226
  101 FORMAT(1H0,10X,31HCYCLE ITERATION FAILED, SEARCH=,I2,4H BU=,F9.0,         CYC 0227
     1      6H EOLK=,F9.5 /)                                                    CYC 0228
C                                                                               CYC 0229
      END                                                                       CYC 0230
```

```
      SUBROUTINE CYEDIT(IC,REGE,IRTYPE)
C
C     EDITS OUT CYCLE INFORMATION
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
      COMMON /EDGE/ NAE(15,16)
      COMMON /CYDEP/ BSHARE(15,10),BBU(10),BKEFF(10),NBBU
C
      DIMENSION REGE(15),IRTYPE(15),KSUR(2,6),RD(15)
C
      DATA KSUR / 4HBURN, 4HUP  , 4HNO. , 3HBPS, 4HEOL , 4HKEFF, 4HFEED,
     1            2H 1, 4HFEED, 2H 2, 4HFEED, 2H 3 /
C
      CALL HEAD
      WRITE (IO6,101) IC
      IS=ISURCH(IC)
      IF (IS.GT.10) IS=IS-7
```

```
      WRITE (IO6,102) (KSUR(I,IS),I=1,2)
      IF (CAPFAC(IC).EQ.0.0.AND.CYTIM(IC).NE.0.0) CAPFAC(IC)=
     1     CYBU(IC)*1.E+5*CYLOAD(IC)/(POWER*CYTIM(IC))
      IF (CAPFAC(IC).EQ.0.0) CAPFAC(IC)=80.
      CYTIM(IC)=CYBU(IC)*CYLOAD(IC)*1.E+5/(POWER*CAPFAC(IC))
      EFPH=0.01*24.*CAPFAC(IC)*CYTIM(IC)
      WRITE (IO6,103) CYBU(IC),EFPH,CYTIM(IC),SDLEN(IC),CAPFAC(IC),
     1           EOLK(IC),BP(IC),CYLOAD(IC)
      NR=NREG(IC)
      WRITE (IO6,104) (I,I=1,NR)
      WRITE (IO6,105) (REGE(I),I=1,NR)
      WRITE (IO6,106) (IRTYPE(I),I=1,NR)
      IF (BP(IC).GT.0.0) WRITE (IO6,107) (BPFRAC(I,IC),I=1,NR)
      WRITE (IO6,108) (NAREG(I,IC),I=1,NR)
      WRITE (IO6,109) (NADIS(I,IC),I=1,NR)
      WRITE (IO6,110) (JSHUF(I,IC),I=1,NR)
      DO 10 I=1,NR
      II=IDXREG(I,IC)
      J1=II/100
      J2=MOD(II,100)
      IF (J1.EQ.1) GO TO 5
      RD(I)=FID(J1,J2)
      GO TO 10
    5 RD(I)=C1ID(J2)
   10 CONTINUE
      WRITE (IO6,111) (RD(I),I=1,NR)
      WRITE (IO6,112) (BOCBU(I,IC),I=1,NR)
      WRITE (IO6,113) (EOCBU(I,IC),I=1,NR)
      WRITE (IO6,114) (I,I=1,NR)
      DO 20 J=1,NBBU
      WRITE (IO6,115) BBU(J),BKEFF(J),(BSHARE(I,J),I=1,NR)
   20 CONTINUE
      DO 25 I=1,16
      DO 25 J=1,15
   25 NAE(J,I)=0
      I1=0
```

```
      IF (IC.GT.1) I1=IEDGE(IC-1)
      I1=I1+1
      I2=IEDGE(IC)
      DO 35 I=I1,I2
      JK=NEDGE(1,I)
      J=JK/100
      K=MOD(JK,100)
      IF (JK.EQ.0) GO TO 35
      IF (J.EQ.0) GO TO 30
      NAE(J,K)=NEDGE(2,I)
      NAE(K,J)=NEDGE(2,I)
      GO TO 35
   30 NAE(K,16)=NEDGE(2,I)
   35 CONTINUE
      WRITE (IO6,116) (I,I=1,NR)
      DO 50 JR=1,NR
      N=0
      DO 40 I=1,16
   40 N=N+NAE(JR,I)
      NN=4*NAREG(JR,IC)
      WRITE (IO6,117) JR,NN,N,NAE(JR,16),(NAE(JR,I),I=1,NR)
   50 CONTINUE
      RETURN
C
  101 FORMAT(30X,5HCYCLE,I3)
  102 FORMAT(1H0,29X,18HSEARCH PARAMETER..,2A4)
  103 FORMAT(1H0,29X,18HBURNUP............,F7.3,8H GWD/MTM /
     1           30X,18HEFPH..............,F7.0,6H HOURS /
     2           30X,18HCYCLE TIME........,F7.1,5H DAYS /
     3                 30X,18HSHUTDOWN LENGTH...,F7.1,5H DAYS /
     3           30X,18HCAPACITY FACTOR...,F7.1, 8H PERCENT /
     4           30X,18HEOL KEFF..........,F7.4  /
     5           30X,18HNUMBER OF BPS.....,F7.0  /
     6           30X,18HLOADING...........,F7.3, 4H MTM )
  104 FORMAT(1H0,37X,10HBATCH DATA / 30X,15I8)
  105 FORMAT(10X,10X,10HENRICHMENT,15F8.3)
```

```
 106 FORMAT(10X,10X,10HFUEL TYPE ,15I8)                                  CYED0109
 107 FORMAT(10X,10X,10HBP FRAC.   ,15F8.3)                               CYED0110
 108 FORMAT(10X,10X,10HASSEMBLIES,15I8)                                  CYED0111
 109 FORMAT(20X,10HDISCHARGED,15I8)                                      CYED0112
 110 FORMAT(20X,10HSOURCE-C,R,15I8)                                      CYED0113
 111 FORMAT(20X,10HREGION ID ,15(4X,A4))                                 CYED0114
 112 FORMAT(20X,10HBOC BURNUP,15F8.3)                                    CYED0115
 113 FORMAT(20X,10HEOC BURNUP,15F8.3)                                    CYED0116
 114 FORMAT(1H0,37X,14HDEPLETION DATA  /10X,10HBOS BURNUP,4X,5HK EFF ,   CYED0117
   1       1X,15I8)                                                      CYED0118
 115 FORMAT(10X,F10.3,F10.4,8F8.3)                                       CYED0119
 116 FORMAT(1H0,19X,9HEDGE DATA /  9X,6HREGION,2X,6HAVAIL.,5X,5HTOTAL,   CYED0120
   1       1X,8HEXTERIOR,1X,15I6)                                        CYED0121
 117 FORMAT(2X,3I10,I8,3X,15I6)                                          CYED0122
C                                                                        CYED0123
     END                                                                 CYED0124
```

```
      SUBROUTINE SOLVE(A,B,EIGEN,N)
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
      REAL*8 A(15,15),B(15),BN(15),EIGEN,SUM,EPS,EIGL,DIF
CDC   REAL    A(15,15),B(15),BN(15),EIGEN,SUM,EPS,EIGL,DIF
C
      DATA MAXIT,EPS,OMEGA /  50,0.00001, 1.80 /, ESP / .001 /
C
      SUM=0.0
      DO 4 I=1,N
    4 SUM=SUM+B(I)
      DO 5 I=1,N
      IF  (SUM.EQ.0.0)  B(I)=1.0/N
      IF  (SUM.GT.0.0)  B(I)=B(I)/SUM
C     IF  (DEBUG)  WRITE  (IO6,102)  (A(I,J),J=1,N),B(I)
    5 CONTINUE
      EIGEN=0.0
```

```
      DO 50 ITER=1,MAXIT
      DO 30 I=1,N
      BN(I)=0.0
      IF (I.EQ.1) GO TO 15
      II=I-1
      DO 10 J=1,II
   10 BN(I)=A(I,J)*B(J)+BN(I)
   15 CONTINUE
      DO 20 J=I,N
      BN(I)=BN(I)+A(I,J)*B(J)
   20 CONTINUE
   30 CONTINUE
      SUM=0.0
      DO 35 I=1,N
   35 SUM=SUM+BN(I)
      DIF=0.0
      DO 40 I=1,N
      BN(I)=BN(I)/SUM
      DIFF=DMAX1(DIF,DABS(BN(I)-B(I)))
CDC   DIFF=AMAX1(DIF, ABS(BN(I)-B(I)))
   40 CONTINUE
      IF (DIF.LE.ESP.AND.DABS(EIGEN-SUM).LE.EPS) GO TO 60
CDC   IF (DIF.LE.ESP.AND. ABS(EIGEN-SUM).LE.EPS) GO TO 60
      EIGEN=SUM
      DO 45 I=1,N
      IF (N.LE.5) GO TO 42
   41 B(I)=B(I)+OMEGA*(BN(I)-B(I))
      GO TO 45
   42 B(I)=BN(I)
   45 CONTINUE
   50 CONTINUE
   60 CONTINUE
      IF (DEBUG) WRITE (IO6,101) ITER,EIGEN
      RETURN
C
  101 FORMAT(10X,5HITER=,I3,3H K=,F9.5)
```

```
  102 FORMAT(10X,10E12.5)                                          SOLV0073
C                                                                  SOLV0074
      END                                                          SOLV0075
```

```
      SUBROUTINE RGEDIT
C
C     SUBROUTINE RGEDIT - EDITS OUT REGION DISCHARGE SUMMARY
C
      COMMON /TABL1/ NBU,NRICH,BURNUP(20),ENRICH(20),ITYPE(20),BIAS(20),
     1               BPWRTH(20),CYBIAS
      COMMON /IO/ IO5,IO6,IO7
      COMMON /TABL2/ KINF(20,20),ISOTOP(20,20,6),POWFIS(20)
      REAL KINF,ISOTOP
      COMMON /BASIC/ TITLE(20),NASS,ANASS,NADIAM,NALPHA,ULOAD,HEIGHT,
     1               BUDIF,ISO,W,ALPHA,ELIM(2),IHAL,POWER,DEBUG
      LOGICAL DEBUG
      COMMON /CYDAT/ ISURCH(21),ISHUF(21),BP(21),CYBU(21),EOLK(21),
     1               NFEED(21),NREG(21),CYLOAD(21),CAPFAC(21),CYTIM(21),
     2               SDLEN(21),IFEDGE(21)
      COMMON /RGDAT/ NEDGE(2,1000),IEDGE(21),NAREG(15,21),JSHUF(15,21),
     1               BPFRAC(15,21),BOCBU(15,21),EOCBU(15,21),
     2               IDXREG(15,21),NADIS(15,21)
      COMMON /CYCL1/ C1RICH(10),C1TYPE(10),C1ID(10),C1LOAD(10),NAC1(10)
      COMMON /FEED / NAFEED(21,3),FTYPE(21,3),FLOAD(21,3),FRICH(21,3),
     1               FID(21,3)
      INTEGER FTYPE,C1TYPE
      COMMON /COST / IFCOST
      COMMON /COEF / COFIT(11),TEMP(2),BWRTH,DPWRTH,XEWRTH
      DIMENSION BBU(10),NAB(10),ICB(10)
      DIMENSION UBU(10),EBU(10),PUBU(10),TBU(10)
C
      NLINE=56
      DO 50 IC=1,21
      IF (ISURCH(IC).EQ.0) GO TO 50
      NF=NFEED(IC)
      IF (IC.EQ.1) NF=NREG(1)
      IF (NF.EQ.0) GO TO 50
      DO 45 IF=1,NF
      NB=0
      IF (IC.GT.1) NAF=NAFEED(IC,IF)
```

```
      IF  (IC.EQ.1)  NAF=NAREG(IF,IC)
      NAF1=NAF
      IF  (IC.EQ.1) GO TO 15
      FRCH=FRICH(IC,IF)
      ITYP=FTYPE(IC,IF)
      RID=FID(IC,IF)
      GO TO 20
   15 CONTINUE
      FRCH=C1RICH(IF)
      ITYP=C1TYPE(IF)
      RID=C1ID(IF)
   20 CONTINUE
      BSUM=0.0
      IDX=100*IC+IF
      DO 10 JC=IC,21
      IF  (ISURCH(JC).EQ.0)  GO TO 10
      NR=NREG(JC)
      DO 5 JR=1,NR
      IF  (IDXREG(JR,JC).NE.IDX)  GO TO 5
      IF  (NADIS(JR,JC).EQ.0)  GO TO 5
      FRAC=NADIS(JR,JC)
      NAF1=NAF1-NADIS(JR,JC)
      NB=NB+1
      NB=MIN0(NB,10)
      ICB(NB)=JC
      NAB(NB)=NADIS(JR,JC)
      FRAC=FRAC/NAF
      DISBU=EOCBU(JR,JC)+BUDIF*(1.0-FRAC)
      BBU(NB)=DISBU
      BSUM=DISBU*NAB(NB)+BSUM
      UBU  (NB)=TERP(DISBU,FRCH,ITYP,ISOTOP(1,1,1),0,KB,BP)
      EBU  (NB)=TERP(DISBU,FRCH,ITYP,ISOTOP(1,1,2),1,KB,BP)
      PUBU(NB)=TERP(DISBU,FRCH,ITYP,ISOTOP(1,1,3),1,KB,BP)
      TBU(NB)=0.0
      IF  (IC.EQ.21)  GO TO 65
      DO 60 I=IC,JC
```

```
60 TBU(NB)=TBU(NB)+CYTIM(I)+SDLEN(I)
   GO TO 75
65 CONTINUE
   N=0
   K=JK
70 CONTINUE
   JK=JSHUF(K,21)
   N=N+1
   K=MOD(JK,100)
   IF (JK.GT.K) GO TO 70
   TBU(NB)=N*(SDLEN(21)+CYTIM(21))
75 CONTINUE
 5 CONTINUE
10 CONTINUE
   IF (NB+1+NLINE.LE.55) GO TO 25
   CALL HEAD
   WRITE (I06,101)
   NLINE=4
25 CONTINUE
   BSUM=BSUM/NAF
   WRITE (I06,102) RID,IF,IC,FRCH,ITYP,NAF,BSUM,ICB(1),NAB(1),BBU(1)
  1      ,TBU(1),UBU(1),EBU(1),PUBU(1)
   NLINE=NLINE+NB+1
   IF (NB.LE.1) GO TO 35
   DO 30 IB=2,NB
30 WRITE (I06,103) ICB(IB),NAB(IB),BBU(IB),TBU(IB),UBU(IB),EBU(IB),
  1      PUBU(IB)
35 CONTINUE
45 CONTINUE
50 CONTINUE
   RETURN
C
101 FORMAT(48X,14HREGION SUMMARY //10X,6HREGION,2X,13HFEED IN CYCLE,
  1      2X,20HENRICH  TYPE  ASMBLY,2X,6HDIS BU,3X,5HCYCLE,2X,
  2      6HASMBLY,3X,6HDIS BU,4X,4HTIME,3X,5HDIS U,2X,6HENRICH,2X,
  3      6HDIS PU)
```

```
  102 FORMAT(1H0,10X,A4,4X,I2,7X,I2,4X,F5.3,3X,I2,4X,I3,4X,F6.3,4X,I2,         RGED0109
     1        6X,I3,4X,F6.3,F8.0,3F8.4)                                         RGED0110
  103 FORMAT(65X,I2,6X,I3,4X,F6.3,F8.0,3F8.4)                                   RGED0111
C                                                                               RGED0112
      END                                                                       RGED0113
```