MIT-3903-4
MITNE-132

SOLUTION OF THE SPACE-DEPENDENT REACTOR KINETICS EQUATIONS IN THREE DIMENSIONS

by
Donald R. Ferguson, K. F. Hansen

August, 1971

Massachusetts Institute of Technology
Department of Nuclear Engineering
Cambridge, Massachusetts 02139

AEC Research and Development Report
Contract AT (30-1) 3903
U.S. Atomic Energy Commission

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF NUCLEAR ENGINEERING

Cambridge, Massachusetts 02139

SOLUTION OF THE SPACE-DEPENDENT REACTOR KINETICS EQUATIONS IN THREE DIMENSIONS

by

Donald R. Ferguson, K. F. Hansen

August, 1971

MIT - 3903 - 4

MITNE - 132

AEC Research and Development Report

Contract AT(30-1) 3903

U.S. Atomic Energy Commission

SOLUTION OF THE SPACE-DEPENDENT REACTOR KINETICS EQUATIONS IN THREE DIMENSIONS

by

Donald Ross Ferguson

Submitted to the Department of Nuclear Engineering on August 16, 1971, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ABSTRACT

A general class of two-step alternating-direction semi-implicit methods is proposed for the approximate solution of the semi-discrete form of the space-dependent reactor kinetics equations. An exponential transformation of the semi-discrete equations is described which has been found to significantly reduce the truncation error when several alternating-direction semi-implicit methods are applied to the transformed equations. A subset of this class is shown to be a consistent approximation to the differential equations and to be numerically stable. Specific members of this subset are compared in one- and two-dimensional numerical experiments. An "optimum" method, termed the NSADE (Non-Symmetric Alternating-Direction Explicit) method is extended to three-dimensional geometries. Subsequent three-dimensional numerical experiments confirm the truncation error, accuracy, and stability properties of this method.

Thesis Supervisor: Kent F. Hansen
Title: Professor of Nuclear Engineering

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Several people and organizations are deserving of an expression of appreciation from the author for the assistance and encouragement they rendered during the course of this work. Professor Kent T. Hansen provided valuable guidance and made himself available for discussions whenever requested by the author.

Professor Allan F. Henry and Dr. William H. Reed shared helpful ideas and observations on several occasions. Much of section 2.4 of this thesis is based on work first completed by Dr. Reed and by Dr. Alan L. Wight.

The author's wife, Signe, provided support and never complained about the countless nights spent by the author at the Information Processing Center.

Fellowship support for the author was provided by the U.S. Atomic Energy Commission. The work was performed under USAEC Contract AT(30-1)-3903. All computation was performed on an IBM 360/65 computer at the M. I. T. Information Processing Center.

Finally, Mrs. Mary Bosco displayed a considerable amount of skill in the typing of this thesis.

BIOGRAPHICAL NOTE

Donald Ross Ferguson was born on May 14, 1944, on a farm near Kensington, Kansas. He attended elementary and secondary school in Kensington, Kansas, and was graduated from Kensington Public High School in May, 1962.

He enrolled at Kansas State University in September, 1962. While an undergraduate, he was a member of FarmHouse social fraternity and served as Chairman of the University's Student Senate and as Vice-President of the Student Body. He was elected to Blue Key National Honor Fraternity. In January, 1967, he was graduated Magna Cum Laude with a B.S. degree in Nuclear Engineering.

After spending seven months as a graduate student at Kansas State University, he entered the University of Birmingham, Birmingham, U.K., as a graduate student in the Department of Physics. He was supported by a Fulbright-Hays Scholarship. He received an M.Sc. degree in Reactor Physics and Technology in December, 1968.

In February, 1969, he entered Massachusetts Institute of Technology as a graduate student in the Department of Nuclear Engineering.

Mr. Ferguson is married to the former Signe Louise Burk of Wichita, Kansas.

Chapter 1

INTRODUCTION

1.1 The Space-Dependent Reactor Kinetics Problem

In the past few years, much effort has been devoted to developing methods for solving the time-dependent multigroup neutron diffusion equations in one or more spatial dimensions. This work has been motivated by at least three reasons. First, it is a mathematical certainty that the solution of these equations for any reactor subjected to a perturbation, which is not homogeneous over the entire reactor, will exhibit a spatially nonuniform behavior. Second, and more practically, the present generation of 1000 Mw(e) and larger light water thermal reactors are so large that they behave in a loosely-coupled manner when subjected to localized perturbations. Finally, the inherently more severe safety problems associated with large liquid-metal-cooled fast breeder reactors must be analyzed as exactingly as possible. Certainly, methods capable of treating space-time effects should be available for use in this analysis.

The time constants associated with the various phenomena which affect the neutron flux distribution in space span many orders of magnitude. Those associated with the burnout of fissile isotopes, buildup of most fission products, and the production of fissile isotopes from fertile isotopes are on the order of weeks and months. Uneven variations in the xenon concentration in space and time can cause spatial power oscillations, with time constants on the order of several hours.

Sodium voiding, loss of coolant in water-cooled reactors, and rapid control rod motions give rise to flux changes with associated time constants on the order of tens of microseconds to a few seconds.

For the most part, those phenomena which occur on a time scale of hours or longer are adequately treated by quasi-static techniques, where the time dependence is treated by a series of static calculations. Of concern for this thesis are methods for treating the more rapid flux variations, where the transient of interest extends over a few seconds at most. The time derivatives cannot be ignored for these transients. These problems are also of the most concern from the standpoint of accident analysis.

For the purposes of this thesis, it is assumed that the multigroup form of the time-dependent diffusion equation is adequate to describe the spatial and energy distribution of the neutron population in a reactor. This is generally true for assemblies of the size of current power reactors, particularly if more exacting methods have been used to obtain the multigroup constants for the various material compositions in the assembly. A more exact mathematical treatment, such as using the time-dependent transport equation, is usually necessary only for more exotic problems such as weapons calculations.

In addition, only the linear form of the multigroup equation is treated in this thesis. Changes in material properties in time are not coupled to local or assembly-wide flux variations. Perturbations are intended to simulate external factors such as control rod motion. Fortunately, for both the reactor designer and for those concerned with methods development, most feedback mechanisms are relatively smooth functions of such factors as temperature and pressure. Since the method developed in this thesis treats problems with time-varying coefficients with no difficulty, it is believed that the method should also treat problems with additional variations in coefficients due to nonlinear feedback effects.

As shown later in this thesis, the equations are finite-differenced on a fixed spatial mesh before they are solved. When the fixed mesh has been specified, an error has been incurred in computing the initial spatial flux distribution and largest eigenvalue when these are compared to the solution of the differential form of the equations. This error is due to the finite mesh spacing. It will be carried on into later time-dependent results obtained from the finite-differenced form of the equations. However, discussions of truncation error in the numerical results shown in this thesis do not refer to this error. Of concern here is the error in the approximate solution when compared to the exact solution of the differential-difference system of equations.

The remainder of this chapter presents the form of the space-dependent reactor kinetics equations to be used hereafter. Several methods previously employed to solve these equations are also reviewed. In Chapter 2, a very general solution technique is derived and shown to possess several desirable and mathematically necessary properties. Four specific variants are considered in more detail for comparative numerical testing. Finally, one of these methods is proposed as being most suitable to three spatial dimensions. Chapter 3 begins with the results of the numerical comparisons of the four variants over a range of problems in one and two spatial dimensions.

Numerical results for four problems in three dimensions, obtained with the method proposed in Chapter 2, are presented to conclude the chapter. Chapter 4 summarizes the conclusions which can be reached concerning this method and includes a discussion of its advantages and limitations.

1.2 The Space-Dependent Reactor Kinetics Equations

The diffusion approximation to the reactor kinetics equations may be written as follows:[1]

$$\frac{1}{v_g}\frac{d\phi_g}{dt}(\vec{r},t) = \vec{\nabla}\cdot D_g(\vec{r},t)\,\vec{\nabla}\phi_g(\vec{r},t) + \sum_{g'=1}^{G}\Sigma_{gg'}(\vec{r},t)\phi_{g'}(\vec{r},t) + \sum_{i=1}^{I} f_{gi}C_i(\vec{r},t) \qquad (1 \leq g \leq G)$$

$$\frac{dC_i}{dt}(\vec{r},t) = -\lambda_i C_i(\vec{r},t) + \sum_{g'=1}^{G} p_{ig'}(\vec{r},t)\phi_{g'}(\vec{r},t) \qquad (1 \leq i \leq I), \tag{1.1}$$

where

g = index number of the energy group

i = index number of the delayed neutron precursor group

ϕ_g = scalar neutron flux in energy group g (neutrons/$cm^2\cdot$ sec)

C_i = density of the i^{th} precursor (cm^{-3})

v_g = speed of the neutrons in the g^{th} group (cm/sec)

D_g = diffusion coefficient for neutrons in group g (cm)

$\Sigma_{gg'}$ = intergroup macroscopic transfer cross section from group g' to group g (cm^{-1}), with the following structure:

$$\Sigma_{gg} = \chi_g \nu_g (1-\beta) \Sigma_{fg} - \Sigma_{ag} - \sum_{g' \neq g} \Sigma_{sg'g} ,$$

χ_g = the fission spectrum yield in group g.

ν_g = average number of neutrons per fission in group g

Σ_{fg} = macroscopic fission cross section in group g

Σ_{ag} = macroscopic absorption cross section in group g

$\Sigma_{sgg'}$ = macroscopic scattering cross section from g′ to g

β = total fractional yield of delayed neutrons per fission.

$$\Sigma_{gg'} = \chi_g \nu_g \Sigma_{fg'} (1-\beta) + \Sigma_{sgg'} , \qquad g' \neq g .$$

$f_{gi} = \lambda_i \chi_{gi}$ = probability (sec^{-1}) that the i^{th} precursor will yield a neutron in group g, where λ_i is the decay constant and $\vec{\chi}_i$ the energy spectrum of neutrons from the i^{th} precursor

$p_{ig'} = \beta_i \nu_{g'} \Sigma_{fg'}$ = production factor (cm^{-1}) for the i^{th} precursor having fractional yield β_i by fissions in group g′.

Boundary conditions for Eqs. (1.1) will be of the homogeneous Neumann or Dirichlet type. At internal interfaces, continuity of the flux and normal component of the neutron current, $\vec{n} \cdot D\vec{\nabla}\phi$, will be required. An initial flux distribution in energy and space must be specified.

Equations (1.1) may be compacted into the form,[1]

$$\frac{d\vec{\theta}}{dt}(\vec{r}, t) = \underline{M}(\vec{r}, t)\, \vec{\theta}(\vec{r}, t) , \tag{1.2}$$

by defining the matrices

$$\vec{\theta}(\vec{r},t) = \begin{bmatrix} \phi_1(\vec{r},t) \\ \phi_2(\vec{r},t) \\ \vdots \\ \phi_G(\vec{r},t) \\ C_1(\vec{r},t) \\ \vdots \\ C_I(\vec{r},t) \end{bmatrix} \tag{1.3a}$$

and

$$\underline{M}(\vec{r},t) =$$

$$\left[\begin{array}{cccc|ccc} v_1(\vec{\nabla}\cdot D_1\vec{\nabla}+\Sigma_{11}) & v_1\Sigma_{12} & \cdots & v_1\Sigma_{1G} & v_1 f_{11} & \cdots & v_1 f_{1I} \\ v_2\Sigma_{21} & v_2(\vec{\nabla}\cdot D_2\vec{\nabla}+\Sigma_{22}) & \cdots & v_2\Sigma_{2G} & v_2 f_{21} & \cdots & v_2 f_{2I} \\ & & \cdots & & & & \\ v_G\Sigma_{G1} & v_G\Sigma_{G2} & \cdots & v_G(\vec{\nabla}\cdot D_G\vec{\nabla}+\Sigma_{GG}) & v_G f_{G1} & \cdots & v_G f_{GI} \\ \hline p_{11} & p_{12} & \cdots & p_{1G} & -\lambda_1 & & 0 \\ & & \cdots & & 0 & \ddots & \\ p_{I1} & p_{I2} & \cdots & p_{IG} & & & -\lambda_I \end{array}\right] \tag{1.3b}$$

This form of the equations will be used later in discussing various mathematical properties of solution techniques proposed in this thesis.

1.3 The Spatially Discretized Equations

Equations (1.1) are continuous in both spatial and temporal variables. In order to discretize the spatial variables, a three-dimensional spatial mesh is superimposed upon the reactor of interest. Equations (1.1) are then integrated over the volumes associated with each of the mesh points, using the box-integration technique.[3] The resulting equations are referred to as the semi-discrete equations.

The semi-discrete forms of the reactor kinetics equations are derived in detail in Appendix A. The resulting equations for the neutron flux at all mesh points for group g, $\vec{\psi}_g$, and the i^{th} precursor concentration at all mesh points, $\vec{C}_i$, can be written as

$$\frac{d\vec{\psi}_g}{dt} = \underline{D}_g \vec{\psi}_g + \sum_{g'=1}^{G} \underline{T}_{gg'} \vec{\psi}_{g'} + \sum_{i=1}^{I} \underline{F}_{gi} \vec{C}_i \qquad (1 \leq g \leq G) \tag{1.4}$$

and

$$\frac{d\vec{C}_i}{dt} = -\underline{\Lambda}_i \vec{C}_i + \sum_{g'=1}^{G} \underline{P}_{ig'} \vec{\psi}_{g'} \qquad (1 \leq i \leq I)\,. \tag{1.5}$$

Here, $\underline{D}_g$ is a seven-stripe matrix representing the net neutron leakage across the six sides of the mesh volume. All other square matrices are diagonal. $\underline{T}_{gg'}$ contains terms representing intergroup transfer processes, and $\underline{F}_{gi}$ represents the transfer of delayed neutrons into group g due to decays in precursor group i. $\underline{\Lambda}_i$ contains the precursor decay constants, while $\underline{P}_{ig'}$ represents the production of delayed precursor i due to fissions in group g′.

Equations (1.4) and (1.5) can be combined into the single matrix equation,

$$\frac{d\vec{\psi}}{dt} = \underline{A}\vec{\psi}. \qquad (1.6)$$

The matrix $\underline{A}$ is square and of order $N * (G+I)$, where N is the number of spatial mesh points. Here, $\vec{\psi}$ and $\underline{A}$ have been defined as

$$\vec{\psi} = \begin{bmatrix} \vec{\psi}_1 \\ \vec{\psi}_2 \\ \vdots \\ \vec{\psi}_G \\ \vec{C}_1 \\ \vdots \\ \vec{C}_I \end{bmatrix} \qquad (1.7)$$

and

$$\underline{A} = \left[\begin{array}{cccc|cccc} \underline{D}_1+\underline{T}_{11} & \underline{T}_{12} & \cdots & \underline{T}_{1G} & \underline{F}_{11} & \cdots & & \underline{F}_{1I} \\ \underline{T}_{21} & \underline{D}_2+\underline{T}_{22} & \cdots & \underline{T}_{2G} & \underline{F}_{21} & \cdots & & \underline{F}_{2I} \\ & & & \cdots & & & & \\ \underline{T}_{G1} & \underline{T}_{G2} & \cdots & \underline{D}_G+\underline{T}_{GG} & \underline{F}_{G1} & \cdots & & \underline{F}_{GI} \\ \hline \underline{P}_{11} & \underline{P}_{12} & \cdots & \underline{P}_{1G} & -\underline{\Lambda}_1 & & & \\ & & & & & & & \underline{0} \\ \underline{P}_{21} & \underline{P}_{22} & \cdots & \underline{P}_{2G} & & \cdot & & \\ & & & \cdots & & & \cdot & \\ & & & & \underline{0} & & & \\ \underline{P}_{I1} & \underline{P}_{I2} & \cdots & \underline{P}_{IG} & & & & -\underline{\Lambda}_I \end{array}\right]. \qquad (1.8)$$

For later reference, several matrices are defined here as follows:

$$
\underline{D} = \left[\begin{array}{cccc|ccc}
\underline{D}_1 & \underline{0} & \cdots & \underline{0} & & & \\
\underline{0} & \underline{D}_2 & \cdots & \underline{0} & & & \\
 & & \cdots & & & \underline{0} & \\
\underline{0} & \underline{0} & \cdots & \underline{D}_G & & & \\
\hline
 & & & & -\underline{\Lambda}_1 & & \underline{0} \\
 & \underline{0} & & & & \ddots & \\
 & & & & \underline{0} & & -\underline{\Lambda}_I
\end{array}\right] , \qquad (1.9a)
$$

$$
\underline{U} = \left[\begin{array}{cccc|ccc}
\underline{0} & \underline{T}_{12} & \cdots & \underline{T}_{1G} & \underline{F}_{11} & \cdots & \underline{F}_{1I} \\
\underline{0} & \underline{0} & \cdots & \underline{T}_{2G} & \underline{F}_{21} & \cdots & \underline{F}_{2I} \\
 & & \cdots & & & & \\
\underline{0} & \underline{0} & & \underline{0} & \underline{F}_{G1} & \cdots & \underline{F}_{GI} \\
\hline
 & \underline{0} & & & & \underline{0} &
\end{array}\right] , \qquad (1.9b)
$$

$$
\underline{L} = \left[\begin{array}{cccc|c}
\underline{0} & \underline{0} & \cdots & \underline{0} & \\
\underline{T}_{21} & \underline{0} & \cdots & \underline{0} & \\
 & & \cdots & & \underline{0} \\
\underline{T}_{G1} & \underline{T}_{G2} & \cdots & \underline{0} & \\
\hline
\underline{P}_{11} & \underline{P}_{12} & \cdots & \underline{P}_{1G} & \\
 & & \cdots & & \underline{0} \\
\underline{P}_{I1} & \underline{P}_{I2} & \cdots & \underline{P}_{IG} &
\end{array}\right] , \qquad (1.9c)
$$

and

$$\underline{T} = \underline{A} - (\underline{D}+\underline{L}+\underline{U}) . \tag{1.9d}$$

For any period of time, Δt, during which all terms in $\underline{A}$ are constant, Eq. (1.6) has the solution

$$\vec{\psi}(\Delta t) = e^{\underline{A}\Delta t}\vec{\psi}(0) . \tag{1.10}$$

All solution techniques for the semi-discrete equations are approximations to Eq. (1.10).

1.4 A Review of Solution Techniques

Calculational methods used for solving the space-dependent kinetics equation can be placed into two broad categories. The first category can be generally classed as modal methods.[4] More specifically, it can be broken into time synthesis and space-time synthesis, both of which could be termed indirect solution techniques. These methods make some assumption about the shape of the solution over several subregions or the entire reactor. These assumptions are forced into the final solution through a variety of techniques. The second category could be termed direct techniques and consists of methods whereby Eqs. (1.1) are solved directly. Since these equations can be solved analytically only for the most trivial of problems, these direct techniques generally involve finite-differencing them and proceeding to solve some approximation to Eq. (1.6).

All of the indirect methods approach the problem by expanding the solution as a linear combination of some set of functions:

$$\vec{\psi}(\vec{r}, t) = \sum_{k=1}^{K} \underline{T}_k(\vec{r}, t)\, \vec{\psi}_k(\vec{r}) . \tag{1.11}$$

The time synthesis methods use one or more $\vec{\psi}_k(\vec{r})$, each of which is defined over the entire solution region. The $\underline{T}_k$ then become functions only of time. The $\vec{\psi}_k(\vec{r})$ may consist of eigenmodes of one of several static operators. Among those suggested are the Helmholtz eigenmodes, the ω-modes, and the λ-modes.[4] None of these have been very successfully applied to any general class of two- or three-dimensional problems.

Alternatively, the $\vec{\psi}_k(\vec{r})$ may be the fundamental modes of a set of operators, each describing the reactor in a different state. Most naturally, these states are chosen to be static states of the reactor at different times during the particular transient of interest.[5] These states can be computed by standard static methods. However, for three-dimensional problems, even the best methods for computing the $\vec{\psi}_k(\vec{r})$ are very time-consuming. It should be noted that the well-known adiabatic method[6] and quasi-static method[7,8] can be considered as variants of time synthesis where only one trial function is used at a time, but new trial functions are used every few time steps.[4]

In space-time synthesis methods, the $\vec{\psi}_k(\vec{r})$ are chosen to represent flux shapes over subregions of the reactor, where the subregion may be a subvolume, plane, or subplane. For example, the so-called single-channel synthesis technique[9,10,11] divides a three-dimensional reactor into a number of axial zones and uses a set of two-dimensional flux shapes for the $\vec{\psi}_k(\vec{r})$ within each zone. The sets may vary from zone to zone, and are chosen to represent static conditions across planes perpendicular to the axis in the zones at various times in the transient. Being only two-dimensional, they are relatively easy to compute. Multi-channel synthesis techniques[12,13] additionally partition the planes perpendicular to the axis into zones and use sets of $\vec{\psi}_k(\vec{r})$ which are allowed to vary independently in these planar zones.

Once the expansion functions have been chosen, equations to be solved for the expansion coefficients are generated using either a variational principle encompassing the multigroup diffusion equations or a weighted residual technique. The great advantage of these methods is that the number of equations to be solved is generally small compared to the number of points at which $\vec{\psi}(\vec{r}, t)$ will be known when the expansion in Eq. (1.11) is carried out, even for three-dimensional calculations. Using a space-time synthesis technique, flux solutions at 10^5-10^6 mesh points over the period of interest in a transient can be obtained in reasonable amounts of computer time.

These synthesis techniques are characterized by a lack of definitive error bounds, however. There is little but intuition to indicate when a set of trial functions will give good results for a particular perturbation.

The direct finite difference techniques, in contrast, are characterized by fairly definitive error estimates. Because of this property, they are extremely useful as numerical standards against which the more approximate methods may be compared. As computational capabilities increase, direct methods also become practical for routine production calculations in one and two dimensions. If fine spatial detail is not required, even three-dimensional direct methods become practical for some types of routine calculations.

In one spatial dimension, the GAKIN[14] and WIGLE[15] methods have been incorporated successfully into codes after which they were named. GAKIN solves Eq. (1.6) by splitting $\underline{A}$ and using the diagonal part of it as an integrating factor to integrate the equation. The behavior of the dependent variables, $\vec{\psi}$, is approximated over each time step so that the integrals can be evaluated.

The WIGLE method approximates the solution to Eq. (1.6) over a series of time steps Δt by

$$\vec{\psi}^{j+1} = \Delta t\, \underline{\theta}\, \underline{A}\, \vec{\psi}^{j+1} + \Delta t\, (\underline{I}-\underline{\theta})\, \underline{A}\, \vec{\psi}^{j}, \tag{1.12}$$

where $\underline{\theta}$ is a diagonal matrix of coefficients, θ_{ii} $(0 \leq \theta_{ii} \leq 1)$. The θ_{ii}'s are chosen to improve the accuracy of the approximation. Setting $\underline{\theta} = \frac{1}{2}\underline{I}$ would yield the Crank-Nicholson approximation with its favorable $O(\Delta t^3)$ truncation error. Thus, relatively large time steps can be taken, but the inversion of the matrix $(I - \Delta t\, \underline{\theta}\, \underline{A})$ must be carried out iteratively. This is equivalent to solving a fixed-source subcritical reactor calculation at each time step.

In two dimensions, the WIGLE method has been extended into the code TWIGL.[16] This code is limited to two neutron groups, but the method could treat any number of groups. Practically, a difficulty arises because even two-group, two-dimensional fixed-source calculations must be done by time-consuming iterative techniques. As more groups are added, time requirements increase rapidly for these iterations.

The LUMAC[17] code extends the GAKIN method to two dimensions by approximating the leakage in first one dimension and then the other by a pointwise transverse buckling over two time steps. The matrices to be inverted at each time step are of the same form as in one dimension and are easily inverted.

Finally, the MITKIN[1,2] method uses a particular alternating-direction, semi-implicit splitting technique referred to as an alternating-direction explicit method.[18] In addition, an exponential transformation is applied to Eq. (1.6), which greatly improves the truncation error. This method is computationally very rapid since all matrices to be inverted are triangular in form. Over a range of problems, it has been shown to be more rapid than the LUMAC algorithm. Increasing the number of mesh points or the number of energy groups results in only a linear increase in computational time. It has also been successfully extended to cylindrical (r-z) and hexagonal geometries.[19]

Motivation for extension of one of these or another method to treat a general class of three-dimensional multigroup problems comes primarily from the need for an accurate numerical standard against which the more rapid synthesis techniques can be tested. In three dimensions, the WIGLE method would be straightforward but extremely time-consuming, due to the great increase in time necessary to perform the three-dimensional, fixed-source-like calculations. Because of its demonstrated superiority over the GAKIN method in two dimensions, the alternating-direction semi-implicit method used in MITKIN is the most promising technique for three dimensions. It is the purpose of this thesis to investigate several variations of this method and extend the "optimum" variation to three dimensions.

Chapter 2

ALTERNATING-DIRECTION SEMI-IMPLICIT TECHNIQUES

It is the purpose of this chapter to examine the theoretical foundations of a class of semi-implicit approximations to the solution of Eq. (1.6), given exactly by Eq. (1.10). Thus, approximations to the operator $\exp(\underline{A}\,\Delta t)$ are examined. Restricting consideration to two-level (first order) approximations of the time derivative, the matrix equivalents of the well-known Padé rational approximations[20] are the most straightforward. Equation (1.12), with $\underline{\theta}$ set to $\underline{0}$, $\underline{I}$, and $\frac{1}{2}\underline{I}$ gives, respectively, the Padé (0, 1), (1, 0), and (1, 1) approximations. However, the (0, 1) approximation suffers from severe stability restrictions,[20] while the (1, 0) and (1, 1) approximations require inversion of a matrix containing $\underline{A}$. This becomes prohibitively time-consuming in problems involving three spatial dimensions and several neutron energy groups.

The class of semi-implicit techniques examined here circumvents this difficulty by "splitting" $\underline{A}$ and inverting only a part of it at a time, a part generally chosen to be easily inverted. The alternating-direction implicit method[21] and alternating-direction explicit method[18] are members of this class. Treating only a part of $\underline{A}$ implicitly necessarily leads to more severe truncation error difficulties and the requirement of much smaller time steps than for methods which invert $\underline{A}$ in its entirety. Thus, application of several of these methods to the direct solution of Eq. (1.6) has been found to be unsatisfactory.[1,22,23]

After reviewing properties of the $\underline{A}$ matrix in section 2.1, an exponential transformation to Eq. (1.6) is introduced in section 2.2. This transformation has been found to significantly reduce the truncation error when several of these "splitting" methods are subsequently applied to the transformed equations.[1,23] Section 2.3 presents a general two-step alternating-direction splitting method for application to the transformed version of Eq. (1.6), and section 2.4 discusses mathematical properties of this method. Four specific splittings of $\underline{A}$ are proposed for further examination in section 2.5. Finally, one of these four is examined in section 2.6 for application to three-dimensional geometries.

2.1 The $\underline{A}$ Matrix

It is instructive to examine the $\underline{A}$ matrix in some detail. The magnitudes of its elements vary over 6 to 8 orders of magnitude. The decay constants λ are on the order of unity, while velocities of order 10^5 to 10^9 multiply absorption and leakage coefficients which may be as large as 10^{-1}. Its eigenvalues likewise span several orders of magnitude, from 10^{-1} sec^{-1} to -10^6 sec^{-1}, giving rise to a property known as "stiffness" to the set of differential equations for reactivities less than prompt critical.[1] Thus, any attempt to represent the derivative in Eq. (1.6) by a finite difference approximation will require that relatively small time steps be taken in order to follow the more rapidly varying components of the solution. At the same time, the interesting part of the transient may span a large number of these time steps.

Additionally, $\underline{A}$ is a real, square irreducible matrix with non-negative off-diagonal elements and negative diagonal elements. In Chapter 8 of Varga,[20] this is termed an "essentially positive" matrix. Varga's Theorem 8.1 states that $\exp(\underline{A}t)$ is positive for all $t > 0$. His Theorem 8.2 further states that $\underline{A}$ has a real, simple eigenvalue, ω_o, which is larger than the real part of any other eigenvalues, ω_i, and to which corresponds a positive eigenvector, $\vec{e}_o$. If any element of $\underline{A}$ increases algebraically, ω_o increases. Finally, his Theorem 8.3 states that the asymptotic behavior of $\exp(\underline{A}t)$ is given by

$$\| \exp(\underline{A}t) \| \sim K \cdot \exp(\omega_o t) \tag{2.1}$$

as $t \to \infty$, where K is some constant, independent of t. This also assumes that $\underline{A}$ is constant. The solution vector $\vec{\psi}(t)$ in Eq. (1.10) will always be non-negative for a non-negative initial condition $\vec{\psi}(0)$. Thus, the desired solution $\vec{\psi}(t)$ is well-behaved and bounded, as physically it must be.

The numerical property of consistency is discussed later in this chapter. The discrete approximation to the $\vec{\nabla} \cdot D\vec{\nabla}$ operator contained in $\underline{A}$ is consistent and accurate to order $(\Delta x)^2$, $(\Delta y)^2$ and $(\Delta z)^2$, the mesh spacings in the three dimensions.[25] Stated in another way, if $\vec{\theta}$ is a genuine solution to Eq. (1.2), then

$$\underline{A}\vec{\theta} = \underline{M}\vec{\theta} + O(\Delta x^2) + O(\Delta y^2) + O(\Delta z^2) \,. \tag{2.2}$$

It is also instructive to observe certain properties of $\underline{D}$ as defined in Eq. (1.9a). Use of the box integration technique to discretize the spatial variables assures that $(-\underline{D})$ is symmetric and diagonally dominant with positive diagonal entries and nonpositive off-diagonal entries.

It is also irreducible. A sufficient condition for $(-\underline{D})$ to be irreducibly diagonally dominant is that homogeneous Dirichlet boundary conditions be specified along at least one of the boundaries. If this is the case, then $\underline{D}$ is negative definite.[3]

2.2 The Exponential Transformation

It is desired to increase the size of the time step size while still controlling truncation error when using alternating-direction splitting methods. A change of variables has been suggested[1,23] which achieves this end. Let

$$\vec{\psi}(t) = e^{\underline{\Omega} t}\,\vec{\phi}(t)\,, \tag{2.3}$$

where $\underline{\Omega}$ is a diagonal matrix of free parameters, henceforth referred to as frequencies. Since $\underline{\Omega}$ is diagonal, the exponential is easily computed.

To obtain an equation for $\vec{\phi}$, differentiate Eq. (2.3) to obtain

$$\frac{d\vec{\psi}}{dt} = e^{\underline{\Omega} t}\,\frac{d\vec{\phi}}{dt} + \underline{\Omega}\, e^{\underline{\Omega} t}\,\vec{\phi}\,. \tag{2.4}$$

Substituting this into Eq. (1.6) yields

$$\frac{d\vec{\phi}}{dt} = e^{-\underline{\Omega} t}\,(\underline{A}-\underline{\Omega})\; e^{\underline{\Omega} t}\,\vec{\phi}\,, \tag{2.5}$$

to be solved for $\vec{\phi}$.

This change of variables has been motivated by the idea that since the behavior of $\vec{\psi}$ is basically exponential in nature, the function $\vec{\phi}$ should be relatively slowly-varying, providing that the $\underline{\Omega}$ are properly chosen. Hence, the time derivative in Eq. (2.5) should be approximated by a simple finite difference with less resultant truncation error than if the same finite difference were used to approximate the time derivative in Eq. (1.6). Equation (2.5) has the same form as does Eq. (1.6), so the same solution techniques are applicable to both.

The choice of $\underline{\Omega}$ is a delicate matter.[1] That such an $\underline{\Omega}$ matrix exists is seen by choosing $\underline{\Omega}$ so that

$$\underline{\Omega}\vec{\psi}(t') = \underline{A}\,\vec{\psi}(t') \,. \tag{2.6}$$

Then

$$\left.\frac{d\vec{\phi}}{dt}\right|_{t=t'} = 0 \,, \tag{2.7}$$

so that in some interval about t′, $\vec{\phi}$ should be slowly varying. For many problems, this interval is long compared to the time step sizes necessary to control truncation error when solving the untransformed equation.

Best results are obtained[1,23] when a new $\underline{\Omega}$ is chosen for each time step, Δt. For the time step from $t = N\Delta t$ to $t = (N+1)\Delta t$, the vector $\vec{\psi}([N+1]\Delta t) = \vec{\psi}^{N+1}$ is not yet known. Using $\vec{\psi}^{N}$ in Eq. (2.6) to compute $\underline{\Omega}$ for this step has been found to be unstable. Providing $\underline{\Omega}$ does not change very much for $t \leq t' \leq t+\Delta t$, it has been found that the Ω values to be used for the neutron groups at point j for this step may be successfully approximated by

$$\left(\Omega^{N}\right)^{\text{group } g}_{\text{point } j} = \frac{1}{\Delta t} \ln \frac{\psi^{N}_{\bar{g},j}}{\psi^{N-1}_{\bar{g},j}} \,, \qquad 1 \leq g \leq G \,. \tag{2.8}$$

All of the groups thus use the same frequency at a mesh point. The group $\bar{g}$ to be used in Eq. (2.8) is the thermal group in thermal reactor problems and a representative fast group in fast reactor problems.

The procedure outlined here can equally well be viewed as an extrapolation procedure. Based on past behavior, the desired solution $\vec{\psi}$ is extrapolated from time t to t+Δt. A relatively small correction factor to this extrapolated behavior is then computed by some finite difference technique. As long as the rate of change of $\vec{\psi}$ is smooth, this extrapolation procedure should work well, thus allowing relatively long time steps to be taken. On the other hand, sudden variations in the rates of change of elements in $\underline{A}$ can cause relatively rapid changes in the behavior of some components of $\vec{\psi}$. When these rapid variations occur, the extrapolation works less well. Smaller time steps must then be taken in order to retain accuracy. This behavior is evidenced in the numerical results shown in Chapter 3.

2.3 A General Two-Step Alternating-Direction Semi-Implicit Method

To apply the general class of alternating-direction splitting methods to Eq. (2.5), the time derivative is replaced by two successive forward differences over a time step, $\Delta t(=2h)$. For notational purposes, let the time step start at t=0 so that $\vec{\psi}(0) = \vec{\phi}(0) = \vec{\phi}^o$. For the two halves of the time step, each of duration h, split $\underline{A}$ as follows:

$$\underline{A} = \underline{A}_1 + \underline{A}_2 \tag{2.9a}$$

and

$$\underline{A} = \underline{A}_3 + \underline{A}_4 . \tag{2.9b}$$

By evaluating the two exponentials at t=h, the midpoint of the step, the difference approximations to Eq. (2.5) become

$$\frac{\vec{\phi}(h) - \vec{\phi}(0)}{h} = e^{-\underline{\Omega}h}(\underline{A}_2 - \alpha\underline{\Omega})\, e^{\underline{\Omega}h}\, \vec{\phi}(h) + e^{-\underline{\Omega}h}(\underline{A}_1 - \gamma\underline{\Omega})\, e^{\underline{\Omega}h}\, \vec{\phi}(0)$$

$$\frac{\vec{\phi}(2h) - \vec{\phi}(h)}{h} = e^{-\underline{\Omega}h}(\underline{A}_4 - \alpha\underline{\Omega})\, e^{\underline{\Omega}h}\, \vec{\phi}(2h) + e^{-\underline{\Omega}h}(\underline{A}_3 - \gamma\underline{\Omega})\, e^{\underline{\Omega}h}\, \vec{\phi}(h)\,, \tag{2.10}$$

where $\alpha + \gamma = 1$.

The unknowns at $t=h$ can be eliminated to yield

$$\vec{\phi}(2h) = e^{-\underline{\Omega}h}[\underline{I} - h(\underline{A}_4 - \alpha\underline{\Omega})]^{-1}[\underline{I} + h(A_3 - \gamma\underline{\Omega})] \cdot$$

$$\cdot [\underline{I} - h(\underline{A}_2 - \alpha\underline{\Omega})]^{-1}[\underline{I} + h(\underline{A}_1 - \gamma\underline{\Omega})]\, e^{\underline{\Omega}h}\, \vec{\phi}(0)\,.$$

Since $\vec{\psi}(2h) = e^{2h\underline{\Omega}}\,\vec{\phi}(2h)$, this can be written as

$$\vec{\psi}(2h) = \vec{\psi}^{\,1} = \underline{B}(\underline{\Omega}, h)\,\vec{\psi}^{\,o}\,, \tag{2.11}$$

where $\underline{B}$ is called the advancement matrix.[1] It is given by

$$\underline{B}(\underline{\Omega}, h) = e^{\underline{\Omega}h}[\underline{I} - h(\underline{A}_4 - \alpha\underline{\Omega})]^{-1}[\underline{I} + h(\underline{A}_3 - \gamma\underline{\Omega})] \cdot$$

$$\cdot [\underline{I} - h(\underline{A}_2 - \alpha\underline{\Omega})]^{-1}[\underline{I} + h(\underline{A}_1 - \gamma\underline{\Omega})]\, e^{\underline{\Omega}h}\,. \tag{2.12}$$

Likewise, for any interval Δt,

$$\vec{\psi}^{\,N+1} = \underline{B}(\underline{\Omega}, h)\,\vec{\psi}^{\,N}\,. \tag{2.13}$$

Equations (2.11) and (2.12) represent an arbitrary alternating-direction semi-implicit method. Although it is termed a two-step method because two successive finite differences are taken to advance the solution over time Δt, it is essential to think of the two operators which advance the solution over each half-step h as inseparable from each other. Either used by itself is quite unstable. However, the error modes most strongly excited by one operator are the ones most strongly damped by the other operator. The solution is thus said to be advanced over one step during time Δt, even though the entire space and energy mesh has been swept twice.

2.4 Properties of the Generalized Method with Transformation

It is imperative to examine the approximation to the solution of Eq. (1.6) given by Eqs. (2.11) and (2.12) with respect to several important numerical properties. This examination has been carried out in a complete and concise fashion in Ref. 1. It is repeated in this thesis for the sake of completeness. The proofs of several theorems and lemmas quoted here are given in Appendix B. The proofs for consistency and stability follow particularly closely those of Ref. 1.

Property 1. Steady State Behavior

For the steady state case where $\underline{A}\vec{\psi}_o = \vec{0}$,

$$\vec{\psi}(2h) = \underline{B}(\underline{0}, h)\vec{\psi}(0) = \vec{\psi}_o , \tag{2.14}$$

which is the exact solution, independent of h. Thus, operation on a $\vec{\psi}_o$ which represents a just-critical configuration by a $\underline{B}(\underline{0}, h)$ formed from an $\underline{A}$ containing the just-critical parameters will result in no change in $\vec{\psi}_o$.

This can be shown by writing Eq. (2.12) with $\underline{\Omega} = \underline{0}$:

$$\underline{B}(\underline{0}, h) = (\underline{I}-h\underline{A}_4)^{-1}(\underline{I}+h\underline{A}_3)(\underline{I}-h\underline{A}_2)^{-1}(\underline{I}+h\underline{A}_1) .$$

Using the splitting relations defined in Eqs. (2.9), this becomes

$$\underline{B}(\underline{0}, h) = (\underline{I}-h\underline{A}_4)^{-1}[\underline{I}-h(\underline{A}_4-\underline{A})](\underline{I}-h\underline{A}_2)^{-1}[\underline{I}-h(\underline{A}_2-\underline{A})] .$$

Since $\underline{A}\vec{\psi}_o = \underline{\vec{0}}$,

$$\underline{B}(\underline{0}, h)\vec{\psi}_o = (\underline{I}-h\underline{A}_4)^{-1}[\underline{I}-h(\underline{A}_4-\underline{A})](\underline{I}-h\underline{A}_2)^{-1}(\underline{I}-h\underline{A}_2)\vec{\psi}_o$$

$$= (\underline{I}-h\underline{A}_4)^{-1}(\underline{I}-h\underline{A}_4)\vec{\psi}_o = \vec{\psi}_o .$$

Property 2. Temporal Truncation Error

This property is concerned with how well the advancement matrix $\underline{B}(\underline{\Omega}, h)$ approximates the exact discrete solution operator $e^{2h\underline{A}}$. For sufficiently small values of h, the difference between the solution computed using $\underline{B}(\underline{\Omega}, h)$ and that computed using $e^{2h\underline{A}}$ over a time step Δt varies approximately as a single power of h. As shown below, for a perfectly symmetric splitting ($\alpha = \gamma = 0.5$, $\underline{A}_1 = \underline{A}_4$, $\underline{A}_2 = \underline{A}_3$), $\underline{B}(\underline{\Omega}, h)$ agrees with the expansion of $e^{2h\underline{A}}$ through terms of order h^2. For any other splitting, the agreement is through terms of order h.

A Taylor series expansion of the exact operator yields

$$e^{2h\underline{A}} = \underline{I} + 2h\underline{A} + 2h^2\underline{A}^2 + \dots \quad . \tag{2.15}$$

Expanding $\underline{B}(\underline{\Omega}, h)$ likewise gives

$$\underline{B}(\underline{\Omega}, h) = \underline{I} + 2h\underline{A} + h^2[(\underline{A}-\underline{\Omega})^2 + 2(\underline{\Omega}\,\underline{A}+\underline{A}\,\underline{\Omega}) + (\underline{A}_4+\underline{A}_2-2\alpha\underline{\Omega})(\underline{A}-\underline{\Omega}) - 2\underline{\Omega}^2] + O(h^3) . \tag{2.16}$$

For the symmetric splitting given above,

$$\underline{B}(\underline{\Omega}, h) = \underline{I} + 2h\underline{A} + 2h^2\underline{A}^2 + O(h^3) . \tag{2.17}$$

For any other splitting, terms of order h^2 remain in Eq. (2.16).

For the approximate solution method outlined here to be most useful, the discrete solution $\vec{\psi}^N$ should approach the exact solution $\vec{\theta}(N\Delta t)$ more and more closely as the spatial and temporal meshs are successively decreased in size. Mathematically, this can be stated as requiring that discrete solutions converge to the solution of the differential equations, Eq. (1.2). A theorem due to Lax[26] enables this convergence to be shown. His theorem states that given a properly posed initial-value problem and a consistent finite-difference approximation, stability is the necessary and sufficient condition for convergence.

It has been found most convenient[1] to carry out proofs of consistency and stability in a Hilbert space L_2. Thus, vector functions $\vec{\theta}(x, y, z, t)$ which are square integrable are to be considered. On this space L_2, the norm of a linear matrix operator $\underline{M}$ is given by

$$\| \underline{M} \| = \sup_{\vec{\theta}} \frac{\| \underline{M}\vec{\theta} \|}{\| \vec{\theta} \|} .$$

It is assumed that Eq. (1.2) with its associated boundary conditions is a properly-posed initial value problem in the space L_2. The consistency and stability of the method proposed are proven here; convergence is inferred from these.

Property 3. Consistency[1]

The domain of the linear operator $\underline{M}$ in Eq. (1.2) is the set of functions $\vec{\theta}(\vec{r})$ which satisfy the appropriate boundary conditions and for which $\vec{\nabla} \cdot D\vec{\nabla}\vec{\theta}$ exists in L_2. Any function $\vec{\theta}(\vec{r}, t)$ which is in this domain for all t in the interval $0 \leq t \leq T$ and which satisfies Eq. (1.2) in the sense that

$$\left\| \frac{\vec{\theta}(\vec{r}, t+h) - \vec{\theta}(\vec{r}, t)}{h} - \underline{M}\vec{\theta}(\vec{r}, t) \right\| \to 0 \text{ as } h \to 0, \qquad 0 \leq t \leq T,$$

is called a genuine solution of the problem.

Informally stated, the consistency condition requires that the temporal finite differencing used to obtain Eq. (2.13) be an approximation to the time derivative of the genuine solution or, equivalently, that

$$\frac{(\underline{B}(\underline{\Omega}, h) - \underline{I})}{2h} \vec{\theta}(\vec{r}, t)$$

be an approximation to $\underline{M}\vec{\theta}(\vec{r}, t)$. How the discrete operator $\underline{B}$ operates on the continuous function $\vec{\theta}$ must be specified. It is assumed that $\underline{B}(\underline{\Omega}, h)$ picks out points from $\vec{\theta}$, and an interpolation rule is applied to the result to make it continuous in space. This interpolation need not be specified for the proofs contained in this thesis.

A more formal statement of the consistency condition is that if, for every $\vec{\theta}$ in the class of genuine solutions whose initial elements $\vec{\theta}(\vec{r}, 0)$ are dense in L_2, the condition[26]

$$\left\| \left[\frac{\underline{B}(\underline{\Omega}, h) - \underline{I}}{2h} - \underline{M} \right] \vec{\theta}(\vec{r}, t) \right\| \rightarrow 0 \text{ as } h \rightarrow 0, \qquad 0 \leq t \leq T,$$

holds, then the operator $\underline{B}(\underline{\Omega}, h)$ is a consistent approximation to the initial-value problem. With the definition of the derivative,

$$\frac{d\vec{\theta}}{dt} = \lim_{h \rightarrow 0} \frac{\vec{\theta}(t+2h) - \vec{\theta}(t)}{2h},$$

the consistency condition may be modified to be

$$\left\| \frac{\vec{\theta}(t+2h) - \underline{B}(\underline{\Omega}, h)\vec{\theta}(t)}{h} \right\| \rightarrow 0 \text{ as } h \rightarrow 0, \qquad (2.18)$$

the form used in the proof of consistency.

The proof[1] begins by factoring $\underline{B}(\underline{\Omega}, h)$ as follows:

$$\underline{B}(\underline{\Omega}, h) = \underline{C}_1(\underline{\Omega}, h) * \underline{C}_2(\underline{\Omega}, h) . \tag{2.19}$$

Here

$$\underline{C}_1(\underline{\Omega}, h) = e^{\underline{\Omega} h}[\underline{I} - h(\underline{A}_4 - \alpha\underline{\Omega})]^{-1}[\underline{I} + h(\underline{A}_3 - \gamma\underline{\Omega})] \tag{2.20a}$$

and

$$\underline{C}_2(\underline{\Omega}, h) = [\underline{I} - h(\underline{A}_2 - \alpha\underline{\Omega})]^{-1}[\underline{I} + h(\underline{A}_1 - \gamma\underline{\Omega})]e^{\underline{\Omega} h} . \tag{2.20b}$$

Lemma 1,[1] stated here and proved in Appendix B, treats the consistency of $\underline{C}_1$ and $\underline{C}_2$.

LEMMA 1. The operators $\underline{C}_1(\underline{\Omega}, h)$ and $\underline{C}_2(\underline{\Omega}, h)$ are consistent.

The only restriction which must be placed on the operator $\underline{B}(\underline{\Omega}, h)$ in order to complete this proof is that as h is decreased, Δx, Δy, and Δz are decreased so that the ratios $h/\Delta x^2$, $h/\Delta y^2$, and $h/\Delta z^2$ are fixed, real constants of any finite size. The need for this restriction is made clear during the discussion concerning the stability of $\underline{B}(\underline{\Omega}, h)$.

Lemma 2,[1] proved in Appendix B, is also necessary for the completion of the consistency proof.

LEMMA 2. If two operators are consistent, then their product is consistent.

With these two lemmas, the consistency proof can be stated in Theorem 1.[1]

THEOREM 1. The difference operator $\underline{B}(\underline{\Omega}, h)$ given in Eq. (2.12) is a consistent approximation.

Lemma 1 has shown that $\underline{C}_1(\underline{\Omega}, h)$ and $\underline{C}_2(\underline{\Omega}, h)$ are consistent. Since their product equals $\underline{B}(\underline{\Omega}, h)$, Lemma 2 provides the proof to this theorem.

Property 4. Stability[1]

In Eqs. (1.9), the matrix $\underline{A}$ has been split into four parts. Of these four, $\underline{D}$ contains all of the terms which relate to the diffusion of neutrons and, in addition, terms relating to precursor decay. In three-dimensional geometries, the first G submatrices, $\underline{D}_g$, on the diagonal have seven nonzero stripes containing terms which are inversely proportional to the square of the mesh spacings Δx, Δy, and Δz. $\underline{D}$ is termed the principle part of $\underline{A}$ as it is the part of $\underline{A}$ which determines the property of stability. This arises because of the requirement that the ratios $h/\Delta x^2$, $h/\Delta y^2$, and $h/\Delta z^2$ be fixed, real constants as h goes to zero. Subsequently, terms in the product $h\underline{D}$ do not vanish as h goes to zero.

For convenience, the matrix $\underline{E}$ is defined as

$$\underline{E} = \underline{E}_1 + \underline{E}_2 = \underline{E}_3 + \underline{E}_4 = \underline{A} - \underline{D}. \tag{2.21}$$

The matrices $\underline{E}_1$, $\underline{E}_2$, $\underline{E}_3$, and $\underline{E}_4$ are those parts of $\underline{E}$ associated with $\underline{A}_1$, $\underline{A}_2$, $\underline{A}_3$, and $\underline{A}_4$, respectively. All terms in $\underline{E}$ are independent of the mesh spacings.

Split $\underline{D}$ according to

$$\underline{D} = \underline{D}_1 + \underline{D}_2. \tag{2.22}$$

Let $\underline{D}_1$ be that part of $\underline{D}$ contained in $\underline{A}_1$ and $\underline{A}_4$ and $\underline{D}_2$ be that part which is contained in $\underline{A}_2$ and $\underline{A}_3$. To complete the proof of stability, it is necessary to restrict the splitting of $\underline{D}$ such that

$$\underline{D}_1 + \underline{D}_1^T \text{ and } \underline{D}_2 + \underline{D}_2^T \text{ are negative definite.}^1 \quad (2.23)$$

As will be seen later, this is not a serious limitation.

From the proof for consistency, it was required that the ratios $h/\Delta x^2$, $h/\Delta y^2$, and $h/\Delta z^2$ be fixed, real constants. Here, those constants are defined as

$$h/\Delta x^2 = \sigma_1 \quad (2.24a)$$

$$h/\Delta y^2 = \sigma_2 \quad (2.24b)$$

$$h/\Delta z^2 = \sigma_3 . \quad (2.24c)$$

The proof for stability examines the case where both the spatial and temporal meshes are taken to zero together. The class of problems where the spatial mesh is fixed and only the temporal mesh is taken to zero is unimportant, because almost any method is stable if h is taken sufficiently small with a given spatial mesh. It is shown that the difference approximation is stable under the conditions of Eqs. (2.24) with σ_1, σ_2, and σ_3 arbitrary and thus is unconditionally stable.[1]

A third condition imposed upon the proof for stability is that all elements in $\underline{A}$ and $\underline{\Omega}$ be held fixed in time. Thus, stability is shown only for each period of time over which this is true. In the algorithm finally used in numerical calculations in this thesis, $\underline{\Omega}$ is changed with each time step Δt. Additionally, elements of $\underline{A}$ may also vary each step, such as during an insertion of reactivity. The much more difficult question of stability for this nonlinear procedure has not been analytically examined yet. Experimentally, however, stability problems have not arisen over a series of sample problems in two- and three-dimensional geometries.

With the difference equations written in the form of Eq. (2.13), a sufficient condition for numerical stability[26] is that

$$\| \underline{B}(\underline{\Omega}, h)^N \| \leq K, \quad K \text{ some constant,}$$
$$0 \leq h \leq \tau, \quad 0 \leq 2Nh \leq T. \tag{2.25}$$

This implies that the computed solution will remain bounded as both spatial and temporal meshes are decreased in size so that more and more steps are required to reach a fixed total time T.

The proof of stability proceeds in several steps. A theorem due to Kreiss and Strang[26] motivates these steps.

THEOREM 2. If the difference system

$$\vec{U}^{N+1} = \underline{C}(\Delta t)\, \vec{U}^N$$

is stable, and if $\underline{Q}(\Delta t)$ is a bounded family of operators, then the difference system

$$\vec{U}^{N+1} = [\underline{C}(\Delta t) + \Delta t\, \underline{Q}(\Delta t)]\, \vec{U}^N$$

is stable.

It thus must first be shown that the operator $\underline{B}(\underline{\Omega}, h)$ can be written as

$$\underline{B}(\underline{\Omega}, h) = \underline{B}'(h) + h\underline{Q}(\underline{\Omega}, h). \tag{2.26}$$

If $\underline{B}'(h)$ can be shown to be stable and $\underline{Q}(\underline{\Omega}, h)$ bounded, then the stability of $\underline{B}(\underline{\Omega}, h)$ is assured.

With $\underline{C}_1(\underline{\Omega}, h)$ and $\underline{C}_2(\underline{\Omega}, h)$ defined as in Eqs. (2.20), $\underline{B}(\underline{\Omega}, h)$ is again factored as

$$\underline{B}(\underline{\Omega}, h) = \underline{C}_1(\underline{\Omega}, h)\underline{C}_2(\underline{\Omega}, h) .$$

The matrix $\underline{C}_1(\underline{\Omega}, h)$ can be factored as

$$\underline{C}_1(\underline{\Omega}, h) = [\underline{I} + h\underline{\Omega} + O(h^2)][\underline{I} - h(\underline{I} - h\underline{D}_1)^{-1}(\underline{E}_4 - \alpha\underline{\Omega})]^{-1} \cdot$$

$$\cdot [\underline{I} - h\underline{D}_1]^{-1}[\underline{I} + h(\underline{D}_2 + \underline{E}_3 - \gamma\underline{\Omega})]$$

$$= [\underline{I} + h\underline{\Omega} + O(h^2)][\underline{I} + h(\underline{I} - h\underline{D}_1)^{-1}(\underline{E}_4 - \alpha\underline{\Omega}) + O(h^2)] \cdot$$

$$\cdot [\underline{I} - h\underline{D}_1]^{-1}[\underline{I} + h(\underline{D}_2 + \underline{E}_3 - \gamma\underline{\Omega})] .$$

Finally,

$$\underline{C}_1(\underline{\Omega}, h) = [\underline{I} - h\underline{D}_1]^{-1}[\underline{I} + h\underline{D}_2] + h\underline{Q}_1(\underline{\Omega}, h) . \qquad (2.27)$$

Similarly, $\underline{C}_2(\underline{\Omega}, h)$ can be written as

$$\underline{C}_2(\underline{\Omega}, h) = [\underline{I} - h\underline{D}_2]^{-1}[\underline{I} + h\underline{D}_1] + h\underline{Q}_2(\underline{\Omega}, h) . \qquad (2.28)$$

Combining Eqs. (2.27) and (2.28) gives

$$\underline{B}(\underline{\Omega}, h) = [\underline{I} - h\underline{D}_1]^{-1}[\underline{I} + h\underline{D}_2][\underline{I} - h\underline{D}_2]^{-1}[\underline{I} + h\underline{D}_1] + h\underline{Q}(\underline{\Omega}, h), \qquad (2.29)$$

so that the matrix $\underline{B}'(h)$ in Eq. (2.26) is defined as

$$\underline{B}'(h) = [\underline{I} - h\underline{D}_1]^{-1}[\underline{I} + h\underline{D}_2][\underline{I} - h\underline{D}_2]^{-1}[\underline{I} + h\underline{D}_1] . \qquad (2.30)$$

Proving the boundedness in the various matrices in $\underline{Q}(\underline{\Omega}, h)$ requires careful analysis. This is because the number of mesh points and, hence, the order of these matrices approach infinity as h is taken toward zero. Theorem 3,[1] the proof of which is given in Appendix B, resolves this issue.

THEOREM 3. A family of matrices $\underline{M}_n$ of varying dimension n having at most $\ell < n$ nonzero elements in each row or column, ℓ being constant for all n, has a uniform L_2 bound if the individual elements of the matrices $\underline{M}_n$ are uniformly bounded for all n.

All elements in $\underline{E}$ and, hence, in $\underline{E}_1$, $\underline{E}_2$, $\underline{E}_3$, and $\underline{E}_4$ are independent of the mesh spacings. Thus they are uniformly bounded. The number of nonzero elements in each row of $\underline{E}$ is less than or equal to the number of prompt and delayed neutron groups. Thus, $\underline{E}_1$, $\underline{E}_2$, $\underline{E}_3$, and $\underline{E}_4$ have uniform L_2 bounds.

The matrix $h\underline{D}$ has at most seven nonzero elements in each row (nine for a hexagonal-z mesh configuration). Providing the conditions given in Eqs. (2.24) are obeyed, the magnitudes of its elements are fixed as h tends toward zero. Thus the L_2 norm of $h\underline{D}$ is bounded for all h. This also assures that $(\underline{I}+h\underline{D}_1)$ and $(\underline{I}+h\underline{D}_2)$ are bounded.

The boundedness of $(\underline{I}-h\underline{D}_1)^{-1}$ and $(\underline{I}-h\underline{D}_2)^{-1}$ is given by Theorem 4, which is proved in Appendix B.

THEOREM 4. The matrices $(\underline{I}-h\underline{R})^{-1}$ and $(\underline{I}+h\underline{R})(\underline{I}-h\underline{R})^{-1}$ have L_2 norms of less than unity provided that $(\underline{R}+\underline{R}^T)$ is negative definite.

All matrices which form the matrix $\underline{Q}(\underline{\Omega}, h)$, as given in Eq. (2.29), have been shown to be bounded. Thus $\underline{Q}(\underline{\Omega}, h)$ is bounded as h tends toward zero. It remains only to show that $\underline{B}'(h)$ is stable. This can be done by factoring it in the form:[1]

$$\underline{B}'(h) = \underline{R}_1\underline{R}_2\underline{R}_3 ,$$

where

$$\underline{R}_1 = (\underline{I}-h\underline{D}_1)^{-1}$$

$$\underline{R}_2 = (\underline{I}+h\underline{D}_2)(\underline{I}-h\underline{D}_2)^{-1}$$

$$\underline{R}_3 = (\underline{I}+h\underline{D}_1) .$$

By Theorem 4, $\|\underline{R}_2\| < 1$ and $\|\underline{R}_3\underline{R}_1\| < 1$. Writing $[\underline{B}'(h)]^N$ in terms of the above factorization,

$$\underline{B}'^N(h) = \underline{R}_1\underline{R}_2\underline{R}_3 \quad \underline{R}_1\underline{R}_2\underline{R}_3 \quad \cdots \quad \underline{R}_1\underline{R}_2\underline{R}_3 \quad \text{(N times)}.$$

Thus,

$$\|\underline{B}'^N(h)\| \leq \|\underline{R}_1\| \cdot \|\underline{R}_2\| \cdot \|\underline{R}_3\underline{R}_1\| \cdot \|\underline{R}_2\| \cdot \|\underline{R}_3\underline{R}_1\| \cdot \ \ldots \ \|\underline{R}_2\| \cdot \|\underline{R}_3\|,$$

$$\|\underline{B}'^N(h)\| < \|\underline{R}_1\| \cdot \|\underline{R}_3\|.$$

Again, $\underline{R}_1$ has a bounded norm by Theorem 4 and $\underline{R}_3$ has a bounded norm by Theorem 3, both for $0 < h < \tau$. Thus, $||\underline{B}'^N(h)||$ is bounded for $0 < h < \tau$ and $0 < 2Nh < T$ and is stable. Finally, from this fact and Theorem 2, $\underline{B}(\underline{\Omega}, h)$ is seen to be stable. Since no restrictions have been placed on the size of σ_1, σ_2, and σ_3 in Eqs. (2.24), except that they be real and finite, this stability is unconditional.

Property 5. Asymptotic Behavior

Because of the form of the exponential transformation, the difference method proposed here can be forced to yield the correct asymptotic behavior. The asymptotic behavior of the exact solution is given by Theorem 5,[24] which is proved in Appendix B.

THEOREM 5. As t approaches infinity, the solution vector $\vec{\psi}(t) = e^{(\underline{A}t)}\vec{\psi}_o$ approaches $\alpha\, e^{\omega_o t}\, \vec{e}_o$, where ω_o is the largest eigenvalue of $\underline{A}$, $\vec{e}_o$ the corresponding eigenvector, and $\alpha = (\vec{\psi}_o, \vec{e}_o)$.

Theorem 6[2] gives the largest eigenvalue and corresponding eigenvector of $\underline{B}(\underline{\Omega}, h)$ under the assumption that $\underline{\Omega} = \omega_o\underline{I}$. It is also proved in Appendix B.

THEOREM 6. If $\underline{\Omega} = \omega_o \underline{I}$, the approximate solution operator $\underline{B}(\underline{\Omega}, h)$ has as its largest eigenvalue $e^{2\omega_o h}$, with corresponding eigenvalue $\vec{e}_o$, where $\underline{A}\,\vec{e}_o = \omega_o \vec{e}_o$.

If, at asymptotic times, the matrix $\underline{\Omega}$ were set equal to $\omega_o \underline{I}$, the action of $\underline{B}(\underline{\Omega}, h)$ on the asymptotic solution would ultimately yield the exact growth of $e^{2h\omega_o}$ over the time step 2h.

2.5 Specific Splittings for Two Dimensions

Up to this point, the splitting of $\underline{A}$ into $\underline{D}$ and $\underline{E}$ and these into $\underline{D}_1$ and $\underline{D}_2$ and $\underline{E}_1$, $\underline{E}_2$, $\underline{E}_3$, and $\underline{E}_4$, respectively, has been very general. Specific splittings must be indicated before proceeding to numerical calculations. Any splitting proposed must obey Condition (2.23) in addition to offering relative computational ease.

Four specific splittings are presented for study in this section. Two of these have been extensively tested previous to this work, the Non-Symmetric Alternating-Direction Explicit (hereafter referred to as NSADE) method in Refs. 1 and 2 and the Symmetric Alternating-Direction Implicit (SADI) method in Refs. 23 and 24. This testing was carried out in two spatial dimensions. The NSADE method has been shown to handle a wide variety of test problems successfully, while the SADI method required unreasonably small time steps to treat a difficult asymmetric problem. The four splittings proposed here for further two-dimensional studies are motivated by a desire to understand what has caused the difference in performance of these two methods and to arrive at an "optimum" splitting.

The terminology used above deserves clarification. The "Symmetric" and "Non-Symmetric" have been prefixed to the names originally given to these methods to indicate the placement of the matrices $\underline{U}$ and $\underline{L}$ in the two splittings of $\underline{A}$. A method is termed symmetric if the matrix $\underline{L}$ is treated implicitly over the first half-step and $\underline{U}$ implicitly over the second half-step. If $\underline{L}$ is treated implicitly over both half-steps, the method is called non-symmetric. If the two-dimensional spatial mesh is swept solving for the new fluxes point by point, the method is termed explicit. It is termed implicit if a whole row or column of points is solved simultaneously for new fluxes.

<u>SADI Method</u>. For this method, let

$$\alpha = \gamma = 0.5$$

$$\underline{A}_1 = \frac{1}{2}\underline{T} + \underline{U} + \underline{D}_1 = \underline{A}_4 \qquad (2.31)$$

$$\underline{A}_2 = \frac{1}{2}\underline{T} + \underline{L} + \underline{D}_2 = \underline{A}_3 ,$$

where $\underline{D}_1$ is composed of the terms associated with diffusion in one direction and one-half of each term in the submatrices $\underline{\Lambda}_i$. The matrix $\underline{D}_2$ is composed of the diffusion terms for the other direction and the remaining half of each term in the $\underline{\Lambda}_i$. As discussed under Property 2, this splitting agrees with the Taylor series expansion of the exact solution operator through terms of order h^2.

NSADI Method. Here let

$$
\begin{aligned}
&\alpha = 1.0\,, \ \gamma = 0\,, \\
&\underline{A}_1 = \underline{U} + \underline{D}_1 \\
&\underline{A}_2 = \underline{T} + \underline{L} + \underline{D}_2 \\
&\underline{A}_3 = \underline{U} + \underline{D}_2 \\
&\underline{A}_4 = \underline{T} + \underline{L} + \underline{D}_1\,,
\end{aligned} \tag{2.32}
$$

where $\underline{D}_1$ and $\underline{D}_2$ are as defined above. By defining the truncation error over one step as

$$\text{T.E.} = e^{2h\underline{A}} - \underline{B}(\underline{\Omega}, h)\,, \tag{2.33}$$

the NSADI method has a truncation error of

$$\text{T.E.} = h^2(\underline{T}+\underline{L}-\underline{U}-\underline{\Omega})(\underline{A}-\underline{\Omega}) + O(h^3)\,.$$

SADE Method. Let

$$
\begin{aligned}
&\alpha = \gamma = 0.5\,, \\
&\underline{A}_1 = \tfrac{1}{2}\,\underline{T} + \underline{U} + \underline{D}_1 = \underline{A}_4 \\
&\underline{A}_2 = \tfrac{1}{2}\,\underline{T} + \underline{L} + \underline{D}_2 = \underline{A}_3\,,
\end{aligned} \tag{2.34}
$$

where $\underline{D}_1$ contains the two stripes of $\underline{D}$ which lie above the diagonal plus one-half of each term on the diagonal and $\underline{D}_2$ contains the two stripes below the diagonal plus the remainder of each diagonal term. As with the SADI method, the truncation error for one time step is of order h^3.

NSADE Method. Let

$$\alpha = 1.0\,,\ \gamma = 0\,,$$

$$\underline{A}_1 = \underline{U} + \underline{D}_1$$

$$\underline{A}_2 = \underline{T} + \underline{L} + \underline{D}_2 \qquad (2.35)$$

$$\underline{A}_3 = \underline{U} + \underline{D}_2$$

$$\underline{A}_4 = \underline{T} + \underline{L} + \underline{D}_1\,,$$

where $\underline{D}_1$ and $\underline{D}_2$ are as defined for the SADE method. Its truncation error is the same as that given for the NSADI method.

It can be seen that all four methods just presented satisfy the conditions for consistency and stability. The box integration technique used to derive the five-point finite difference relations in two dimensions guarantees that the diagonal term in each row of $\underline{D}$ is just the negative of the sum of the other terms in that row. Both implicit and explicit splittings make the diagonal term in each row in both $\underline{D}_1$ and $\underline{D}_2$ the negative sum of the other two terms in that row. Thus, both $\underline{D}_1$ and $\underline{D}_2$ are diagonally dominant. Since

$$\underline{D}_1 + \underline{D}_1^T = \underline{D}_2 + \underline{D}_2^T = \underline{D}$$

for both splittings and $\underline{D}$ is negative definite, the condition (2.23) is satisfied.

All four methods offer relative computational ease. The matrices to be inverted in the SADE and NSADE methods are always upper or lower triangular or can be made so by rearranging the order of the unknowns. The first half-step is carried out by forward substitution, sweeping from one corner of the mesh to the diagonally-opposite corner and from the highest energy group to the lowest. The second half-step reverses the direction of the spatial sweep and also from the lowest energy group to the highest in the case of the SADE method.

For the SADI and NSADI methods, the matrices to be inverted are block lower or upper triangular, but the diagonal submatrices are tri-diagonal. In sweeping from one corner of the mesh to the diagonally opposite corner, entire lines of fluxes in one of the two directions must be solved simultaneously by the rapid forward elimination, backward substitution process. In working back across the mesh during the second half-step, lines of fluxes in the second direction are solved simultaneously. For the NSADI method, the groups are solved from the highest to the lowest energy over both half-steps, while the order is reversed for the second half-step of the SADI method.

This section is concluded with a discussion of the factors which could cause these four methods to perform differently on actual numerical experiments. The first difference apparent is the implicit versus explicit spatial treatment. From experience gained in static calculations, it is tempting to state that solving for an entire line of fluxes simultaneously should result in less total error than solving for the fluxes one by one. The analogy is not entirely appropriate, however, since the kinetics problem is an initial-value problem and not a boundary-value problem. Considering the two sweeps of the mesh together, new fluxes at each of the five points in two-dimensional problems are given half of the weighting and old fluxes the other half for both types of methods.

There does appear to be a difference in the spatial distribution of the errors for the two spatial treatments. No analytical examination of error distribution and propagation has yet been completed. Qualitatively, however, experience seems to indicate that the implicit treatment is somewhat more stable with respect to propagation of errors.

For illustrative purposes, consider the first time step, Δt, in a two group homogeneous problem, where the initial condition $\vec{\psi}_o$ is taken to be exact. Let the perturbation be due to uniform step decreases in the absorption cross sections of both groups over the entire system.

Both the implicit and explicit methods are inexact so that some error is introduced into the new group one flux as it is calculated at each mesh point over the first half-step. This error is distributed differently for the two methods, however. In the implicit treatment, each line of fluxes is computed simultaneously, using the old fluxes on each side of it to compute the leakage in the direction perpendicular to that line. Thus, the error in the growth is distributed along the entire line. The new fluxes in other lines see none of the error introduced in that line. At the end of the mesh sweep for group one, it is easily shown that the error at each mesh point is proportional to the initial flux value at that point for this model problem.

The group two fluxes at the end of the first half-step likewise contain an error component which has the same spatial distribution as the initial fundamental mode solution. Part of the error at each point is due to error in the group one flux previously computed, and part is due to inexact treatment of the growth of group two given the group one flux.

At the end of the second half-step, additional errors have been introduced into both group fluxes at each mesh point. However, the errors still have a fundamental mode distribution for both groups. No spatial flux tiltings have been introduced by the implicit spatial treatment.

This is not the case with the explicit spatial treatment. As group one fluxes are computed one by one over the first mesh sweep, the error introduced at a point due to the inexact operator is carried on across to the computation of all subsequent mesh points. At the end of the first sweep for group one, the spatial distribution is tilted so that the last point calculated has grown proportionately more than any point previously computed.

If this were a one group problem, the tilting would be erased as the sweep is reversed over the second half-step. In the two group problem, however, the second group must first be calculated. The second group now sees a tilted source and is tilted proportionately worse at the end of the first mesh sweep.

This tilted second group is used in computing the source for the reverse mesh sweep for group one. It is difficult to predict exactly how the group one flux will be distributed at the end of the reverse sweep since that depends on the reactor size and composition and the magnitude of the initial perturbation. However, it would be strictly fortuitous if the errors in the group one flux have a fundamental mode distribution. The first mesh sweeps for the two groups have introduced higher error modes which tend to persist in the solution, although the stability proof in section 2.4 gives assurance that they will not grow without bound for the case of constant $\underline{\Omega}$ and reactor properties.

The really important question is to what degree does the introduction of these higher error modes affect the solution of real problems. In actual practice, it has been found that for realistic perturbations and time step sizes, these higher modes do not severely affect the solution. In addition, the exponential transformation tends to damp out these higher modes, as is shown in the numerical results given in Chapter 3.

There is one situation, however, in which this accumulation of errors can severely hamper the explicit methods. If the initial condition $\vec{\psi}_o$ used to start the transient differs sufficiently from the true fundamental mode initial condition, the presence of these additional errors can affect a sufficient accumulation of error to swamp the true solution.

It should be noted that a fully explicit method cannot properly treat the fluxes at an outer boundary where a zero current normal to that boundary has been specified. This problem was noted in the initial work done in extending the NSADE method to r-z geometry,[19] where the z-axis is always a so-called symmetry boundary. It is easily solved, however, by solving for new fluxes at each point on such a boundary and the interior point closest to it simultaneously for whichever of the two half-steps originates from that boundary.

A second difference to be noted in the methods is the symmetric versus non-symmetric sweeping of the energy groups. Favoring the symmetric methods is the fact that terms of order h^2 in the truncation error expression vanish for these splittings. On the other hand, most thermal reactor models have group structures which are closely coupled by down-scattering from each group to the next lowest, but are only loosely coupled by the upward flow of neutrons. This is because the higher energy groups have relatively small fission cross sections, while the fission spectrum is nonzero only in the highest groups. During a sweep of the energy mesh from the lowest energy group to the highest group, a perturbation in the thermal group can cause a change only in the high energy groups with nonzero fission fractions during the remainder of that sweep. In a two group thermal reactor problem, this effect should be minimal. With four or more groups, this effect could become important. This effect should also be minimized in a fast reactor problem, where the fission cross section is fairly constant over most of the groups, and the fission spectrum is nonzero over most of the groups.

The concept of truncation error accumulation is complicated by the presence of the exponential transformation. It is generally stated that the total error at time $T=2Nh$ varies as a function of one order less of h than does the local truncation error. The correct asymptotic behavior resulting from the exponential transformation should tend to lessen the severity of error accumulation, however.

2.6 A Proposed Method for Three Dimensions: NSADE

It is the stated purpose of this thesis to develop an alternating-direction semi-implicit method for solving the space-dependent kinetics equations in three-dimensional geometries. The method so proposed is the NSADE (Non-Symmetric Alternating-Direction Explicit) method as outlined in section 2.5. The splitting of the $\underline{A}$ matrix for three dimensions is identical to that presented in Eq. (2.35) for two dimensions. However, $\underline{D}_1$ now has three nonzero stripes above the diagonal and $\underline{D}_2$ has three nonzero stripes below it. Because the $\underline{L}$ matrix is treated implicitly over both half-steps, the groups are always to be solved from the highest energy group to the lowest. The spatial sweep starts in one corner of the three-dimensional mesh and works toward the diagonally-opposite corner during the first half-step. It is then exactly reversed for the second half-step.

This particular method has been chosen for three reasons. Based on a number of test problems in one and two dimensions, it is shown in Chapter 3 that the non-symmetric splittings perform far more satisfactorily in thermal reactor problems. Secondly, the NSADI and NSADE methods are shown to perform practically identically over a range of problems. Finally, in addition to being computationally slightly faster, the NSADE method is directly applicable to three-dimensional geometries as a two-step method. Only two dimensions could be treated implicitly if an implicit method as outlined in section 2.5 were to be applied to three-dimensional geometries as a two-step method.

Chapter 3

NUMERICAL RESULTS

Four different members of a general class of alternating-direction semi-implicit methods for solution of the semi-discrete reactor kinetics equations have been proposed in section 2.5 for further study in one- and two-dimensional geometries. The results of several numerical experiments, where these methods have been used to solve reactor problems, are presented and compared in section 3.1 of this chapter. In section 3.2, the behavior of the NSADE method when solving a three-dimensional model problem is compared to the exact solution of this problem. Finally, section 3.3 presents the results of a number of true space-dependent, three-dimensional numerical experiments with the NSADE method.

3.1 One- and Two-Dimensional Studies

Two of the four specific methods that are presented in section 2.5 have been extensively tested previous to this thesis. The NSADE method has been shown to perform satisfactorily over a range of problems in x-y, r-z, and hexagonal geometries.[1,19] In contrast, the SADI method has been shown to perform poorly in a space-dependent, four group thermal reactor problem.[23] The numerical experiments presented in this section have been performed in an effort to explain the difference in behavior of these two methods.

Four different test cases are examined in this section. They have been chosen in an attempt to compare the methods over a wide range of problem types. The first three cases are in one-dimensional slab geometry, while the fourth is the two-dimensional rectangular multi-region thermal reactor which the SADI method had difficulty in treating.

In order to solve the one-dimensional problems, the computational subroutines of an existing one-dimensional code, GAKIN,[14] were replaced with a single subroutine which, depending on several input parameters, treated problems with one of the four methods. Since one-dimensional problems have diffusion on one direction only, the diffusion terms in that direction were halved, with one-half of each term in the matrix $\underline{D}$ being treated as diffusion in one dimension and the other half as diffusion in a second dimension. For the two-dimensional case, subroutines were added to the code MITKIN[1] so that it had multi-method capabilities.

Both because it is the primary purpose of this thesis to deal with multi-dimensional geometries and because the one-dimensional problems treated for this thesis are relatively simplistic, the three one-dimensional problems are discussed here in a qualitative fashion only. The numerical results are not presented in either tabular or graphical form.

The first one-dimensional problem was a homogeneous thermal slab reactor with four neutron groups and one precursor group. The critical configuration was perturbed by a fifty-cent step insertion of reactivity caused by uniformly decreasing the thermal group capture cross section. Twenty-one mesh points were used to represent the

146-cm slab. Because of the homogeneous composition, the initial flux distribution in each group was cosinusoidal in shape. The exact solution to the time-dependent problem was obtained using an eigenvector expansion technique[2] and was available for comparison.

Using a time step, Δt, of .0005 sec, both the SADI and SADE methods underestimated the solution throughout the transient. At 1.0 seconds into the transient, both solutions were about 15 % too low. With Δt = .00025 sec, both methods gave considerably better results, but were still about 1% low at 1.0 sec. Only when Δt was reduced to .0001 sec did the SADI method give the correct result (< .1% error) throughout the transient. The SADE method was not used at this small time step since it was expected that it would again behave similarly to the SADI method.

In contrast, both the NSADE and NSADI methods gave good results (< .1% error) for time steps as large as Δt = .001 sec out to about 0.2 sec into the transient. At around 0.2 sec, however, both methods were overcome by stability problems for time steps of .001 and .0005 sec. The instabilities seemed to result from the feedback of accumulated errors through the frequencies. These instabilities first appeared as a small ripple-like component superimposed on the true solution, but soon grew to the point that negative fluxes resulted.

The characteristic which separated the four methods into two distinct classes is the property which has been termed symmetry. The symmetric methods behaved in one fashion, while the non-symmetric methods behaved in another and different fashion.

These results shed light on several of the conjectures made in section 2.5 about these methods. The group structure for this four group problem was loosely-coupled by the upward flow of neutrons, thus causing the symmetric methods to underpredict the growth of the fluxes at time steps reasonable for this problem. This tendency to underpredict can also be explained from an analytic point of view. In the limit of large h, the advancement matrix goes to the identity matrix for the symmetric methods. For any finite time step, the symmetric methods underpredict the growth over each time step. The feedback effect introduced by the method used to compute the frequencies may offset this to some extent, but the numerical experiment cited here indicates that it does not offset it completely. Once a sufficiently small time step is used, however, these methods converge rapidly to the correct solution.

The non-symmetric methods, even though they have a local truncation error of only order h^2, followed the solution closely for much large time steps. Physically, this smaller error at each step was the result of sweeping down through the groups at both half steps, taking advantage of the tightly-coupled downward flow of neutrons. The instabilities observed prove that these methods can also become unstable due to the feedback effect of the frequencies. Fortunately, these instabilities have never been noted in problems in two or three dimensions or in one-dimensional problems with a large number of mesh points.

The second one-dimensional problem was a homogenized slab unit cell, 10 cm in width, from a fast gas-cooled reactor with ten neutron groups and four precursor groups. The initial flux distribution was flat for all groups. The critical configuration was perturbed by a step reduction in the capture cross sections in all groups.

Only the two implicit methods could be used to treat this problem because the explicit options were not programmed to handle homogeneous Neumann boundary conditions. The SADI method followed the transient accurately for as long as the solution was carried out, although relatively small time steps had to be taken. Physically, this problem was better suited to the symmetric techniques because the fission cross section was fairly constant over most of the groups, and the fission spectrum was nonzero over most of the groups. Thus, even though there was no upscattering in this problem, a perturbation could propagate in an upward sweep of the groups as well as in a downward sweep.

The NSADI method followed the early part of the transient as well as did the SADI method, using the same time step sizes. However, at about .0005 seconds into the transient, instabilities again appeared and soon swamped the true solution. A close examination reveals one reason why these non-symmetric methods should be more susceptible to these feedback-induced instabilities. Unlike the symmetric methods, the non-symmetric methods have advancement matrices which do not reduce to the identity matrix in the limit of large time steps. Depending on the problem and the flux vector at a particular time, they can underestimate or overestimate the flux at the end of the next time step. Add to this the feedback effect of the method used to compute the frequencies, and it becomes possible for these oscillations to grow large. Again it is stressed that these instabilities have been observed only in one-dimensional problems with a relatively small number of spatial mesh points.

The last one-dimensional problem used to compare these four methods was a 240-cm, three-region thermal slab reactor with two neutron groups and six precursor groups. An inner zone, 160 cm thick, of relatively low enrichment, was surrounded on either side with a 40-cm-thick slab of higher enrichment. Ninety-seven equally-spaced mesh points were used. The critical configuration was perturbed by linearly decreasing the thermal capture cross section by 1% over 1.0 second in one of the two outer slabs.

The composition of this test problem was similar to a graphite slab reactor, so that a relatively large time step, Δt, of .0025 second was used. Both the NSADI and NSADE methods followed the transient out to 1.0 second with little error and with no sign of any instabilities. As in the first test case, the SADI and SADE methods initially underestimated the growth in the solution. However, they both improved considerably by the end of the transient.

For two-group problems such as this, the two groups are tightly coupled by both the upward and downward flow of neutrons. This apparently minimized the difference in performance between the symmetric and non-symmetric methods for this problem. Again, the method used to sweep the spatial mesh made little difference in the results.

The final numerical experiment discussed in this section is a highly-asymmetric, two-dimensional problem with four neutron groups and one delayed precursor group. This problem has been discussed in two previous works,[1,23] but it is included here because it again demonstrates the validity of the arguments presented in section 2.5.

The geometry for this problem was identical to that of any plane perpendicular to the z-axis taken between z mesh planes 8 and 17 of Configuration 3, found in Appendix C. The material constants for the four materials were also identical to those shown in Configuration 3, except that the critical value of ν for all groups was 1.450679 for the two-dimensional problem. The critical configuration was perturbed by linearly decreasing the group four capture cross section in material 4 by 0.003 cm^{-1} over 0.2 second. From that time, all material properties were held fixed.

Tables 3.1, 3.2, 3.3, and 3.4 list the group one and group four fluxes at two points in the reactor for various times in the transient. The results for the SADI method have been taken from Ref. 24, while the NSADE results represent improved results (more accurate initial flux distribution) of those quoted in Ref. 1.

The NSADE and NSADI methods gave practically identical results for the results shown, with Δt = .001 sec. Results using the NSADE method and time steps of .0005 sec and .002 sec gave similar results to those listed here, so it is assumed that the results for the two non-symmetric methods represent converged solutions. In contrast, the SADI method gave inconsistent results for time steps as small as .00025 sec and was still nearly 6% in error at 0.3 sec into the transient with a Δt = .000125 sec.

This problem represented a severe test of these methods because of the large changes in the spatial shape and energy spectrum induced by the perturbation. The results shown here again confirm that the method used to sweep the spatial mesh makes little difference in the final result. For thermal reactors, the critical factor is that the groups be swept from high energy to low energy over both half steps. The non-symmetric methods are thus preferred for any scheme which is to have general applicability.

Table 3.1. Group 1 Fluxes at Point (3, 9)

Time		NSADE	NSADI	SADI		
(sec)	Δt =	.001	.001	.000125	.0005	.001
.0		.4463	.4463	.4463	.4463	.4463
.05		.4561	.4559	.4525	.4463	.4463
.10		.4670	.4669	.4781	.4464	.4463
.15		.4796	.4795	.4985	—	.4463
.20		.4943	.4944	.5064	.4624	.4463
.30		.4945	.4946	.5194	.4624	.4465

Table 3.2. Group 1 Fluxes at Point (12, 3)

Time		NSADE	NSADI	SADI		
(sec)	Δt =	.001	.001	.000125	.0005	.001
.0		.1341	.1341	.1341	.1341	.1341
.05		.1383	.1383	.1375	.1346	.1342
.10		.1431	.1430	.1473	.1371	.1346
.15		.1485	.1485	.1554	—	.1359
.20		.1549	.1549	.1604	.1413	.1382
.30		.1549	.1550	.1640	.1489	.1438

Table 3.3. Group 4 Fluxes at Point (3, 9)

Time		NSADE	NSADI	SADI		
(sec)	Δt =	.001	.001	.000125	.0005	.001
.0		.0359	.0359	.0359	.0359	.0359
.05		.0367	.0367	.0364	.0359	.0359
.10		.0376	.0376	.0385	.0360	.0359
.15		.0386	.0386	.0401	—	.0359
.20		.0398	.0398	.0408	.0361	.0359
.30		.0398	.0398	.0418	.0373	.0360

Table 3.4. Group 4 Fluxes at Point (12, 3)

Time		NSADE	NSADI	SADI		
(sec)	Δt =	.001	.001	.000125	.0005	.001
.0		.9684	.9684	.9684	.9684	.9684
.05		1.0532	1.0528	1.0474	1.0255	1.0223
.10		1.1513	1.1510	1.1855	1.1006	1.0873
.15		1.2669	1.2668	1.3278	—	1.1614
.20		1.4051	1.4051	1.4565	1.2914	1.2498
.30		1.4060	1.4064	1.4920	1.3498	1.2889

3.2 Three-Dimensional Studies: Homogeneous Problem

As stated in section 2.6, the NSADE method has been chosen as the method to be extended to treat three-dimensional geometries. Four numerical experiments have been designed to test this method. The geometries and compositions for these experiments are presented in Appendix C. The results from the first of these, the only homogeneous problem, are presented in this section. All of the numerical results from three-dimensional experiments have been obtained from a computer code called 3DKIN, which is discussed in Appendices D and E.

Again, it must be stressed that the truncation error discussed in this chapter is the difference between the particular solution under consideration and the exact solution of the semi-discrete equations. In the case of the homogeneous problem, the exact solution can be generated using an eigenvector expansion technique.[2] The exact solutions cannot be obtained for the other three-dimensional problems. Thus it is assumed that if two successive solutions are generated, one using a time step half of the size of that used to generate the other, and are in good agreement, then the solution generated with the smaller time step represents a "converged" solution.

TEST CASE 1

Geometry and Composition: Configuration 1

Perturbation: Step change, $\Delta\Sigma_a$ (group 2) = $-.369 \times 10^{-4}$

This case is a bare, homogeneous cube, 200 cm on a side, with two neutron groups and one precursor group. Ten mesh intervals were used in each direction, and the boundary conditions were homogeneous Dirichlet on all six sides. The perturbation consisted of a uniform step decrease in the thermal group absorption cross section and had a reactivity worth of about 50 cents. Since the geometry is symmetric about the mid-plane in the x-direction, only the right half of the reactor was actually used in the 3DKIN computer runs. It was determined that the half-core and full-core results compared through six significant figures for two different time step sizes.

The results of 3DKIN runs using four different time step sizes at various times in the transient are shown in Table 3.5. The values presented are the thermal group fluxes at the center point of the reactor.

Table 3.5. Test Case 1 Results, Group Two Fluxes at Centerpoint

Time		3DKIN				EXACT
(sec)	$\Delta t =$	.01	.005	.002	.001	
.0		.816	.816	.816	.816	.816
.05		.920	1.043	1.116	1.124	1.127
.10		1.151	1.361	1.403	1.406	1.407
.15		1.454	1.651	1.660	1.660	1.660
.20		1.782	1.904	1.892	1.890	1.890
.30		2.388	2.328	2.294	2.289	2.288
.40		2.840	2.671	2.628	2.622	2.620

Table 3.5 demonstrates the rapid convergence of the NSADE method with the exponential transformation. With a time step of .002 sec, the solution was only .3% in error at .4 second, during which time the thermal flux had more than tripled. That this convergence is approximately of order h^2 is displayed in Fig. 3.1, where the percentage truncation error is plotted as a function of h at 0.4 second into the transient.

The results that are tabulated in Table 3.5 are presented in graphical form in Fig. 3.2 to illustrate an interesting characteristic of this exponentially-transformed method. The semi-discrete equations are a coupled set of first-order differential equations. As such, any change in $\vec{\psi}$ at time t depends only on the values of $\vec{\psi}$ and $\underline{A}$ at that time

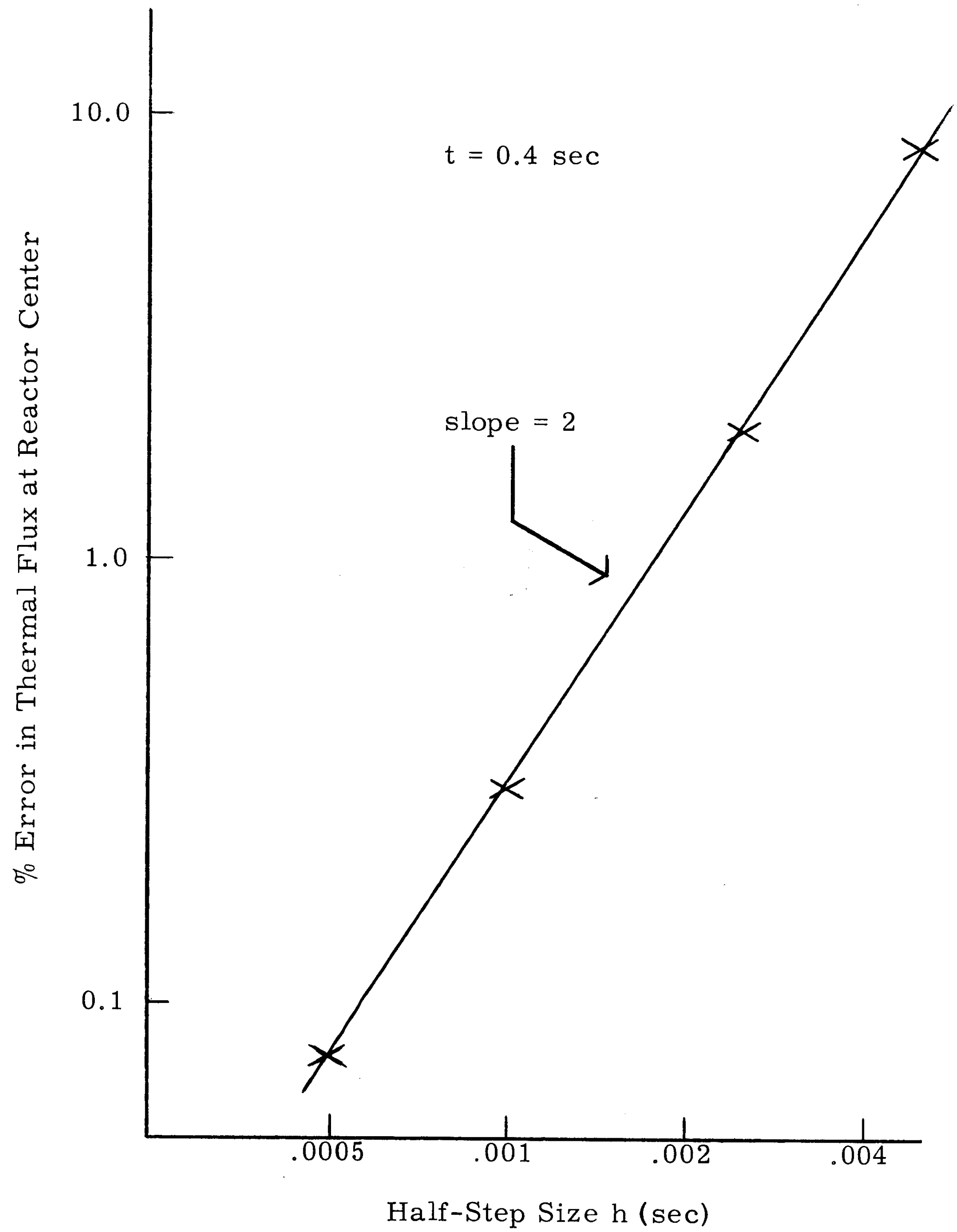

Fig. 3.1. Convergence Rate for Test Case 1

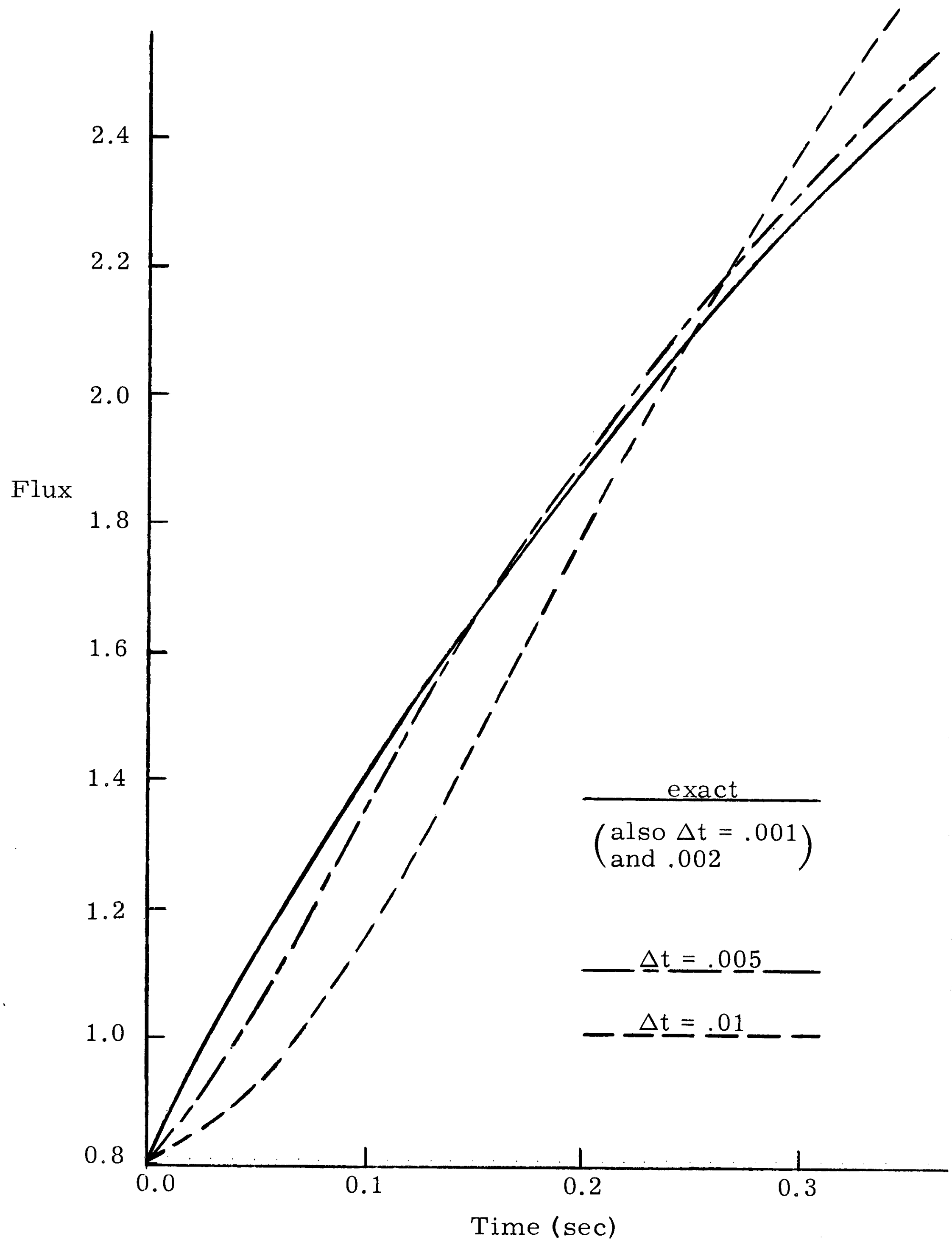

Fig. 3.2. Test Case 1 Results, Centerline Thermal Flux

and not on the past history of the system. By adding the exponential transformation and computing $\underline{\Omega}$ to be used at t_N based on the change in the solution between t_{N-1} and t_N, the behavior of the solution at t_N has been coupled to its rate of change. The system now behaves in the fashion of a second order system in that it builds up "inertia" during a transient. Figure 3.2 clearly displays the damped sinusoidal oscillations superimposed on the true solution which are characteristic of such a system. The amplitude of the "overshoot" is clearly a function of h and decreases as order h^2.

When material properties are constant or changing in a smooth fashion, this "inertia" enables the time step to be increased without affecting the accuracy seriously. However, when properties or their rates of change are abruptly changed, such as at the end of a ramp insertion of reactivity, time step sizes must be decreased in order to overcome the "inertia."

3.3 Three-Dimensional Studies: Space-Dependent Problems

The three test cases presented in this section are all spatially-dependent problems. Test Cases 2 and 3 are three-dimensional versions of problems already used to test some or all of the methods discussed in section 2.5. Test Case 4 is a new problem, designed with the idea of simulating the withdrawal of a cluster of control rods from two adjacent subassemblies in a medium-sized pressurized-water power reactor. Taken together, these problems provide a stern test of the general applicability of the NSADE method.

TEST CASE 2

Geometry and Composition: Configuration 2

Perturbation: Ramp change, $\Delta\Sigma_a$ (material 1, group 2) = -.0045(t/0.2)

for $0 \leq t \leq 0.2$ sec

$\Delta\Sigma_a$ (material 1, group 2) = -.0045

for $t > 0.2$ sec

The original two-dimensional version of this problem has been used to test several two-dimensional solution methods.[1, 23, 17, 15] The original plane was 160 cm square, with a central blanket area surrounded by a highly-enriched seed area. It was in turn surrounded by another blanket region. In three dimensions, this configuration was made 112 cm thick in the z-direction, and a blanket of 24 cm thickness added to the top and bottom. Thus, the overall reactor is cubical, 160 cm on a side. It has two neutron groups and one delayed precursor group.

The four regions containing material 1, each 32 × 32 × 124 cm in size, which were perturbed are located symmetrically with respect to the central x-plane. Only the right half of the cube was considered, with a homogeneous Neumann boundary condition imposed at the exposed mid-plane to preserve symmetry. With 8.0-cm mesh spacings in each direction, a total of 4841 mesh points were needed to represent the half-reactor. The initial flux distribution and eigenvalue were computed with the steady state option of 3DKIN.

Test Case 2 was carried out to 0.3 second into the transient for time step sizes of .001 sec and .002 sec. The results of the 3DKIN runs for these two time step sizes are presented in Tables 3.6, 3.7, and 3.8. The values tabulated are the thermal flux values. The z-planes 3 and 19 are 8 cm below and above the core, respectively, while z-plane 11 is the central z-plane. Point (6, 6, z) is on the central z-axis of one of the perturbed regions. Values at points (6, 16, z) are not shown in these tables. However, they agreed with corresponding values at points (6, 6, z) to better than 0.05% for every z value, thus preserving symmetry.

Table 3.6. Test Case 2 Results, z-Plane 3

Time		Point (1, 11, 3)		Point (6, 6, 3)	
(sec)	Δt =	.002	.001	.002	.001
.0		.347	.347	.245	.245
.05		.392	.398	.280	.284
.10		.484	.483	.350	.349
.15		.610	.619	.446	.454
.20		.853	.867	.633	.643
.25		1.094	.994	.811	.737
.30		.998	.991	.740	.735

Table 3.7 Test Case 2 Results, z-Plane 11

Time		Point (1, 11, 11)		Point (6, 6, 11)	
(sec)	Δt =	.002	.001	.002	.001
.0		1.279	1.279	.422	.422
.05		1.442	1.467	.487	.496
.10		1.784	1.780	.617	.616
.15		2.248	2.284	.796	.809
.20		3.149	3.197	1.144	1.162
.25		4.035	3.666	1.465	1.330
.30		3.679	3.655	1.334	1.326

Table 3.8. Test Case 2 Results, z-Plane 19

Time		Point(1,11,19)		Point(6,6,19)	
(sec)	Δt =	.002	.001	.002	.001
.0		.347	.347	.245	.245
.05		.388	.395	.278	.283
.10		.480	.479	.348	.348
.15		.605	.615	.444	.452
.20		.847	.860	.630	.640
.25		1.086	.987	.808	.734
.30		.991	.984	.737	.732

The results at a Δt of .001 indicate that the thermal flux grew by factors of 2.86 and 3.16 at the reactor center and in the center of the perturbed regions, respectively. The group one fluxes grew by practically equal amounts. Thus, spatial and energy spectral changes were minimal, as would be expected for this symmetric problem.

From Tables 3.6 and 3.8, it is seen that differences of up to 1% exist in the results at planes 3 and 19, when they should be equal. After these runs were made, an error was discovered in 3DKIN which caused several coefficients for points on z-plane 18 to be incorrectly computed. This was the cause of the slight retardation in growth in z-plane 19 flux values. With the error corrected, a later run was carried out to .10 sec and gave results symmetric to 4 significant figures in the z-direction. The runs shown here were not repeated because of the cost of the 2-1/2 hours of computer time required to do so.

At $\Delta t = .002$ sec, the solution considerably overshot the true solution during time $0.2 \leq t \leq 0.3$ sec. To overcome this damped oscillatory behavior when the run with $\Delta t = .001$ sec was made, the time step was decreased to .0005 sec for .01 sec just as the ramp was cut off. This was largely successful as the solution then overshot by only a very small amount. Closer examination of the solution at several times in the range $0.2 \leq t \leq 0.3$ revealed that the peak of the overshoot occurred at .25 sec and that the solution was growing smoothly and asymptotically by $t = 0.3$ sec. It is believed that the solution shown here for $\Delta t = .001$ sec has converged to less than 1% error at all times except perhaps at the peak of the overshoot. A run made out to .10 sec with $\Delta t = .0005$ sec supported this statement for that part of the transient.

TEST CASE 3

Geometry and Composition: Configuration 3

Perturbation: Ramp change, $\Delta\Sigma_a$ (material 4, group 4) = $-.0035(t/0.2)$ for $0 \leq t \leq 0.2$ sec

$\Delta\Sigma_a$ (material 4, group 4) = $-.0035$ for $t \geq 0.2$ sec

As mentioned in section 3.1, this problem, with four neutron groups and one precursor group, is a three-dimensional version of a problem used to compare several methods in two dimensions. Specifically, the original 160 cm × 80 cm plane was made 120 cm thick in the z-direction. However, the bottom 56 cm of the region with material 4 was changed to material 3 (which was identical to material 4 before the perturbation). Thus, only the top 64 cm was perturbed for this transient.

This problem is asymmetric in all three dimensions so that the full reactor with homogeneous Dirichlet boundary conditions had to be considered. With 8.0-cm mesh spacings, a total of 3696 mesh points were used.

Material 1 is a highly enriched material so that group one fluxes were initially more than five times higher than group four fluxes in it. On the other hand, materials 3 and 4 are strong moderators so that the group four flux peaked in them. Given these spectral variations in the initial condition, which was computed with 3DKIN, and the asymmetric perturbation, it was expected that large spatial and energy spectrum changes would result.

The results of runs made on 3DKIN out to 0.3 second with time step sizes of .002 and .001 sec are shown in Tables 3.9 through 3.12. Point (3, 9, z) is near the center of the highly-enriched core, while point (12, 3, z) is in the center of the perturbed region for $z > 56$ cm. z-plane 4 is the mid-plane of the unperturbed lower portion, while z-plane 12 is near the center of the upper 64 cm region.

As expected, this transient resulted in rather severe spectral changes. At point (3, 9, 4), the group one and group four fluxes grew by only 6%. Meanwhile, the group one and group four fluxes at point (12, 3, 12) grew by 11% and 45%, respectively. The solution overshot slightly at the end of the ramp for $\Delta t = .002$ sec, but practically all traces of overshoot were wiped out during the run with $\Delta t = .001$ sec.

Table 3.9. Test Case 3 Results, z-Plane 4, Group 1

Time		Point (3, 9, 4)		Point (12, 3, 4)	
(sec)	Δt =	.002	.001	.002	.001
.0		1.402	1.402	.384	.384
.05		1.416	1.419	.389	.390
.10		1.439	1.438	.396	.396
.15		1.457	1.459	.402	.403
.20		1.487	1.484	.411	.410
.25		1.487	1.484	.411	.411
.30		1.484	1.486	.410	.410

Table 3.10. Test Case 3 Results, z-Plane 4, Group 4

Time		Point (3, 9, 4)		Point (12, 3, 4)	
(sec)	Δt =	.002	.001	.002	.001
.0		.112	.112	2.742	2.742
.05		.114	.114	2.775	2.781
.10		.115	.115	2.825	2.824
.15		.117	.117	2.867	2.872
.20		.119	.119	2.931	2.928
.25		.119	.119	2.935	2.930
.30		.119	.119	2.928	2.934

Table 3.11. Test Case 3 Results, z-Plane 12, Group 1

Time		Point (3, 9, 12)		Point (12, 3, 12)	
(sec)	Δt =	.002	.001	.002	.001
.0		1.772	1.772	.486	.486
.05		1.791	1.795	.496	.497
.10		1.821	1.820	.510	.509
.15		1.845	1.848	.522	.523
.20		1.883	1.881	.539	.538
.25		1.885	1.881	.539	.538
.30		1.881	1.883	.538	.539

Table 3.12. Test Case 3 Results, z-Plane 12, Group 4

Time (sec)	$\Delta t =$	Point (3, 9, 12) .002	Point (3, 9, 12) .001	Point (12, 3, 12) .002	Point (12, 3, 12) .001
.0		.142	.142	3.467	3.467
.05		.144	.144	3.755	3.764
.10		.146	.146	4.114	4.112
.15		.148	.148	4.510	4.521
.20		.151	.151	5.010	5.008
.25		.151	.151	5.026	5.012
.30		.151	.151	5.012	5.019

The results at the two time step sizes are in good agreement and are thought to represent a good approximation to the exact solution.

TEST CASE 4

Geometry and Compositions: Configuration 4

Perturbation: Ramp changes, $\Delta\Sigma_a$ (material 5, group 2) = $-.004\ (t/.08)$

for $0 \leqslant t \leqslant 0.08$ sec

$\Delta\Sigma_a$ (material 5, group 2) = $-.004$

for $t \geqslant 0.08$ sec

$\Delta\Sigma_a$ (material 6, group 2) = 0

for $0 \leqslant t \leqslant 0.08$ sec

$\Delta\Sigma_a$ (material 6, group 2) = $-.004\left(\frac{t-.08}{.08}\right)$

for $0.08 \leqslant t \leqslant 0.16$ sec

$\Delta\Sigma_a$ (material 6, group 2) = $-.004$

for $t > 0.16$ sec

(continued)

$$\Delta\Sigma_a \text{(material 7, group 2)} = 0$$
$$\text{for } 0 \leqslant t \leqslant 0.16 \text{ sec}$$

$$\Delta\Sigma_a \text{(material 7, group 2)} = -.004\left(\frac{t-.16}{.08}\right)$$
$$\text{for } 0.16 \leqslant t \leqslant 0.24 \text{ sec}$$

$$\Delta\Sigma_a \text{(material 7, group 2)} = -.004$$
$$\text{for } t \geqslant 0.24 \text{ sec}$$

This problem represents an attempt to simulate the withdrawal of control rods from two adjacent subassemblies in a medium-sized pressurized-water power reactor with two neutron groups and one precursor group. The central core zone consists of 16 square subassemblies, each 30 cm on a side, containing 2.8% enriched U^{235}. Four subassemblies of the same size, but containing 3.3% enriched U^{235}, are located along each side of the inner zone. The four 30-cm-square corners plus a 20-cm-thick band around the entire reactor consist of a water and steel reflector. The active core height is 240 cm, with a reflector of 30-cm thickness located above and below it.

The two subassemblies which were perturbed were adjacent to each other with the x mid-plane passing between them. Thus, only the right half of the reactor was considered for the computer calculations. A spatial mesh with 13 $\times$ 25 $\times$ 20 mesh points was used. A homogeneous Neumann boundary condition was imposed on the exposed mid-plane of the reactor.

The rod withdrawal was simulated by linearly decreasing the thermal absorption cross section over three successive time zones of 0.08 sec length. During the first zone, only the bottom third of the subassembly was perturbed. The middle and upper thirds followed successively in the next two zones. With the full perturbation inserted, the reactor had about fifty cents of excess reactivity.

The thermal group fluxes at three heights in the core, both in the center of the perturbed subassembly and in the center of the sub-assembly located symmetrically across the y mid-plane from it, are tabulated in Tables 3.13 through 3.15. Runs were made on 3DKIN with time steps of .002 and .001 sec. The results for $\Delta t = .001$ sec are also plotted on Figs. 3.3 and 3.4.

Table 3.13. Test Case 4 Results, z-Plane 5

Time		Point (1, 5, 5)		Point (1, 21, 5)	
(sec)	Δt =	.002	.001	.002	.001
.0		.291	.291	.291	.291
.04		.296	.299	.364	.369
.08		.313	.313	.492	.493
.12		.330	.337	.556	.567
.16		.376	.381	.684	.694
.20		.439	.415	.803	.768
.24		.439	.442	.819	.828
.28		.466	.456	.879	.857
.32		.463	.457	.870	.859
.35		.453	.458	.850	.861

Table 3.14. Test Case 4 Results, z-plane 10

Time (sec)	Δt =	Point (1, 5, 10)		Point (1, 21, 10)	
		.002	.001	.002	.001
.0		.547	.547	.547	.547
.04		.552	.559	.570	.577
.08		.581	.579	.625	.624
.12		.615	.625	.821	.838
.16		.696	.706	1.212	1.236
.20		.816	.773	1.473	1.401
.24		.824	.828	1.544	1.557
.28		.874	.855	1.660	1.616
.32		.868	.857	1.642	1.619
.35		.850	.859	1.604	1.623

Table 3.15. Test Case 4 Results, z-Plane 16

Time (sec)	Δt =	Point (1, 5, 16)		Point (1, 21, 16)	
		.002	.001	.002	.001
.0		.291	.291	.291	.291
.04		.292	.297	.294	.298
.08		.306	.305	.309	.308
.12		.324	.328	.345	.349
.16		.365	.369	.416	.422
.20		.431	.407	.606	.581
.24		.438	.441	.806	.820
.28		.466	.456	.877	.858
.32		.462	.457	.868	.858
.35		.453	.458	.851	.860

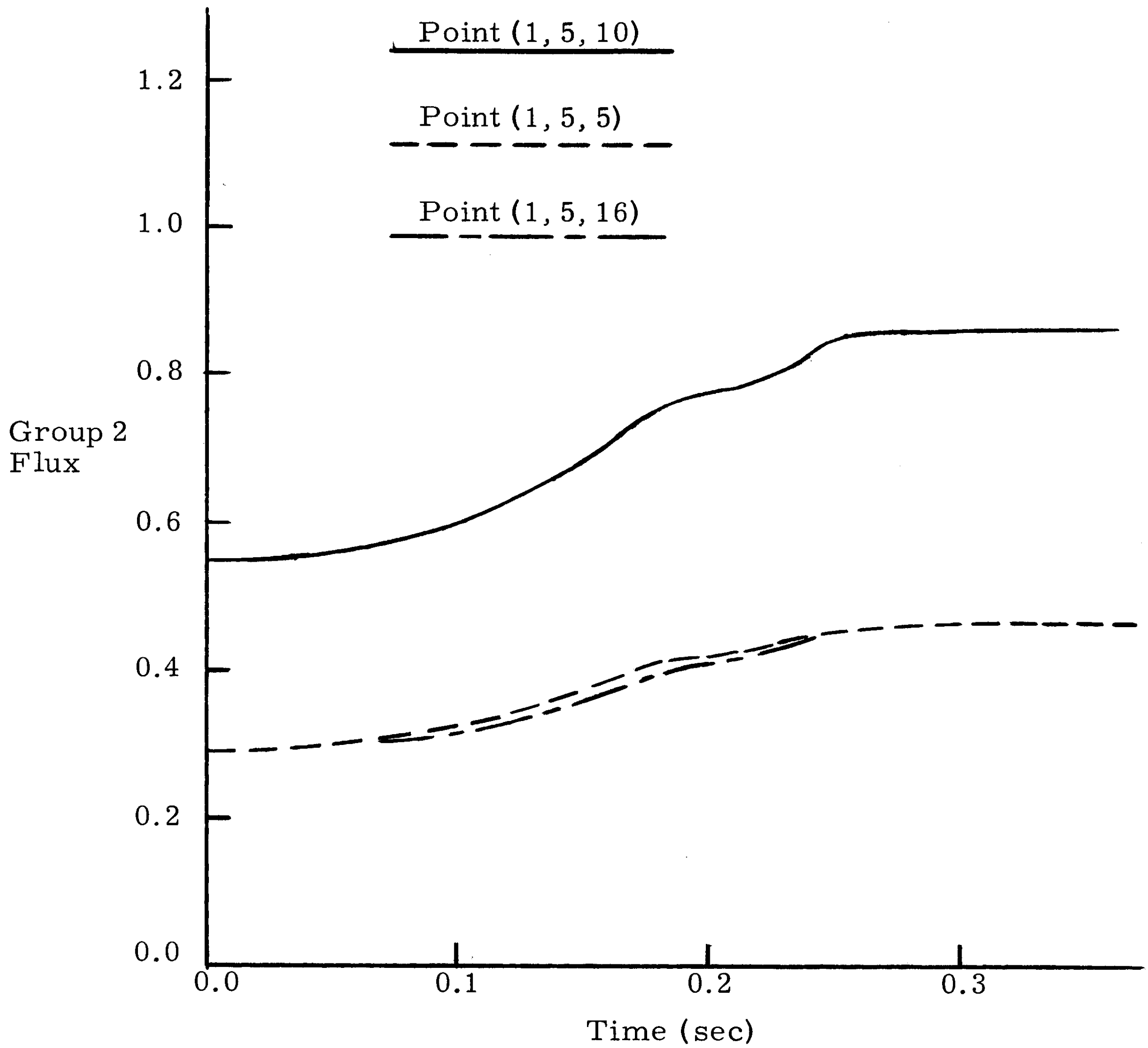

Fig. 3.3. Test Case 4 Results, Point (1, 5, z)

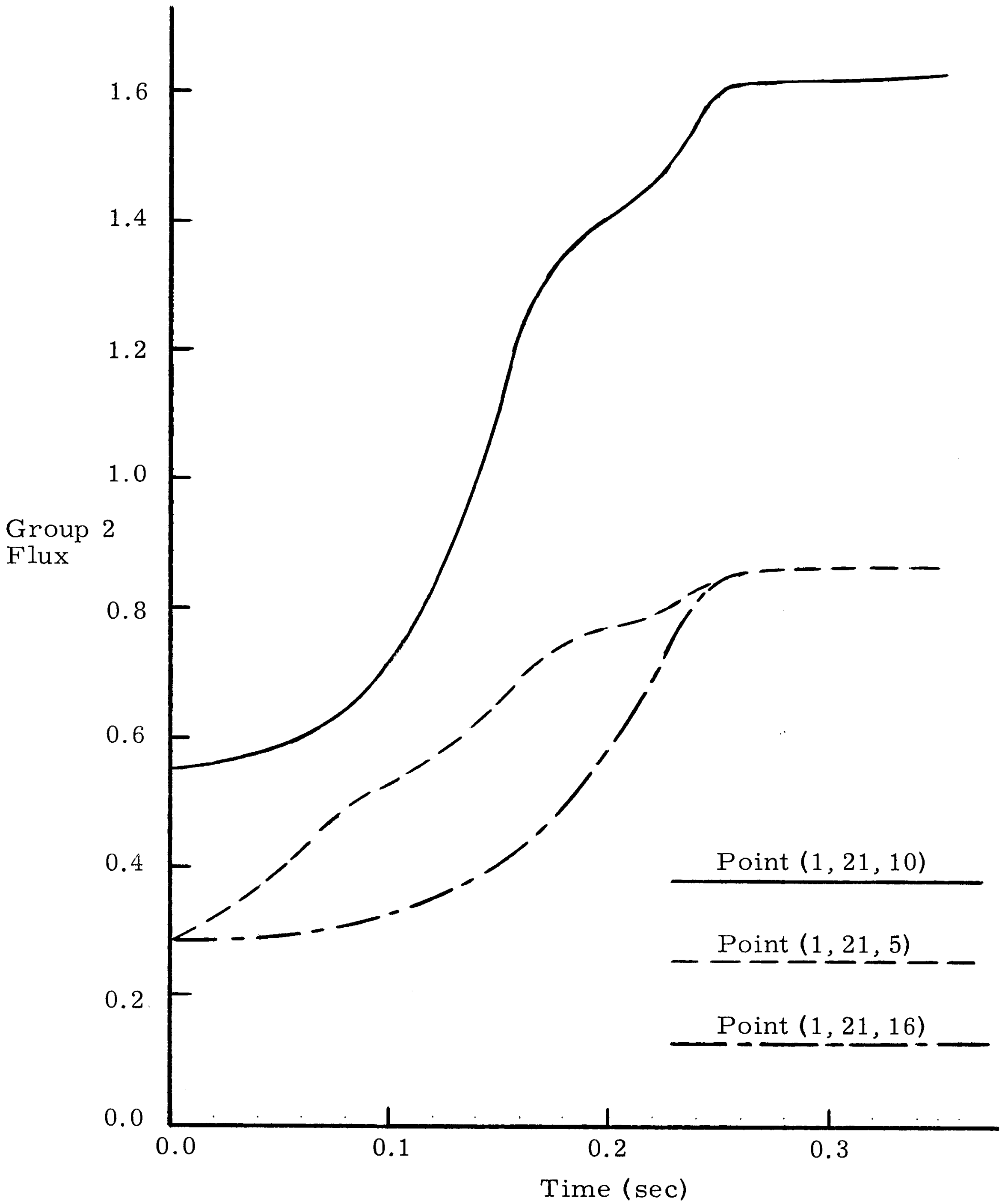

Fig. 3.4. Test Case 4 Results, Point (1, 21, z)

As expected, this perturbation caused severe flux tilting in the reactor. The flux at point (1, 21, 10) grew by a factor of 2.97 during .35 sec, while the flux at point (1, 5, 10) grew by only a factor of 1.57. Likewise, the flux in the upper portion of the core lagged that in the lower third considerably early in the transient, but caught up nicely within..10 sec after the perturbation had become symmetric in the z-direction.

As in earlier cases, the solution at $\Delta t = .002$ behaved in a damped oscillatory fashion at the end of the ramp, due to the frequency calculation. Again, these oscillations disappeared when Δt was halved to .001 sec and halved again for .02 sec just after the end of the ramp. Based on the smoothness of the solution with $\Delta t = .001$ sec and the relatively good agreement of the two solutions except at the end of the ramp, it is again believed that the solution obtained with $\Delta t = .001$ sec is a good approximation to the exact solution.

Chapter 4

CONCLUSIONS AND RECOMMENDATIONS

To be a truly useful numerical technique, a proposed method must treat difficult, practical problems successfully with reasonable computational costs, as well as possess desirable analytical properties. It has been concluded in section 2.4 that the NSADE method satisfies certain analytical criteria necessary for success. This chapter summarizes the practical experience gained from the several numerical experiments presented in Chapter 3.

4.1 Characteristics of the Numerical Results

Several important characteristics are easily observed in the numerical experiments. The property of truncation error behavior for the NSADE method has been shown to be approximately of order h^2, as predicted by the theoretical analysis, for the one problem where it could be accurately measured.

Closely related to truncation error is accuracy. Over several test cases, the NSADE method has been seen to give acceptably accurate solutions at reasonable time step sizes. It is unfortunate that solutions with even smaller Δt's are not available for Test Cases 2, 3, and 4 to further verify the accuracy of the solutions shown. Given the relatively slow computer available for numerical experiments for this thesis, this was just too costly.

It is granted that the time steps required by the NSADE method are probably an order of magnitude smaller than those which would be required for similar accuracy by a direct solution technique where the $\underline{A}$ matrix is not split before inversion. However, it is difficult to imagine any such method, which would necessarily require an iterative technique to carry out the inversion process, requiring less than an order of magnitude more computational effort per time step.

The time step size used by the NSADE method is limited by two factors. These generally come into play during different parts of a transient. During that part of the transient where reactivity is being inserted, usually by an externally-controlled factor such as control rod motion, the time steps are initially limited by the rate of reactivity insertion. This is necessary so that truncation error is controlled while the frequencies used in the exponential transformation are "seeking" the rates of flux change in the various regions of the reactor. Once this has happened, the time steps can be gradually increased in size with little effect on accuracy, so long as the rate of reactivity change remains fairly constant.

During any part of the transient when the rate of reactivity change is suddenly altered, the time steps must be decreased in size. This is necessary if accuracy is to be retained while the frequencies again "seek" the new rates of flux change. This must be done to control the damped oscillations that arise if a time step too large is used through this part of the transient.

A rule of thumb which was first offered for the NSADE method in two dimensions[1] and which has been found to hold approximately for three dimensions relates the truncation error to the rate of solution change over one time step. A 1% change in the solution over each time step generally produces about 1% error in about 100 steps. For a given problem, this implies that about 100 steps are required to predict a doubling in the flux to 1% accuracy.

The numerical stability of the NSADE method has never been found to be a limiting factor in two- and three-dimensional calculations. The oscillations which plague the solution during periods of abrupt change in the rate of reactivity change affect the accuracy of the solution temporarily. They quickly damp out, however, so that the solution returns to the correct rate of change. This correct asymptotic behavior is a result of the exponential transformation. The time step sizes to be used for a particular problem are thus primarily limited by the accuracy desired in the solution.

One great advantage of the NSADE method is its computational ease. All matrix inversions required by it are simple back-substitutions. Because of this, computational times per time step for a range of problems vary approximately linearly with the number of mesh points and neutron groups. It has thus been found possible[1] to derive an expression of the form

$$\text{Time/Step} = \alpha N(G + \beta I) ,$$

which relates the time necessary to advance the solution over one time step, Δt, to the number of unknowns in the problem. Here, N is the number of mesh points in the problem, and G and I are the number of neutron and precursor groups, respectively.

Listed in Table 4.1 below are running times per step required by 3DKIN for four different problems. The computer used for these runs was an IBM 360/65 running under OS/360-MVT. All unknowns were stored in fast memory.

Table 4.1. Computational Times

Mesh Points	Groups	Precursors	Seconds/Step
1331	2	1	3.09
3696	4	1	16.0
4851	2	1	13.3
6500	2	1	18.3

Since only problems with one precursor are available, a value of $\beta = 0.3$ will be used as determined in previous two-dimensional work. From Table 4.1, two values of α are obtained:

$$\alpha = 1.2 \times 10^{-3} \quad \text{for } G = 2$$

$$\alpha = 1.0 \times 10^{-3} \quad \text{for } G = 4 .$$

As G increases, the work per group decreases in 3DKIN since only one frequency is computed for all neutron groups at each mesh point.

4.2 Applicability of the NSADE Method

The numerical experiments presented in Chapter 3 offer strong evidence that the NSADE method is capable of treating a general class of transients in three spatial dimensions with reasonable time step sizes. These include difficult sub-prompt critical transients which result in significant spatial flux tilting and energy spectrum changes.

It is obvious that it would not be feasible to solve problems of a really practical size with the computer which was used for the numerical experiments for this thesis. Table 4.2 compares the floating-point add time (for 64-bit words) of the IBM 360/65 to those of several of the fastest computer systems currently in use or being installed. An extrapolation from the relation developed in the last section for the IBM 360/65 should be approximately correct if it is based on the information in the table.

Table 4.2. Comparison of Computing Speeds

Computer Model	Floating Point Add Time (microseconds)
IBM 360/65	1.8
CDC 6600	0.4
IBM 370/195	0.11
CDC 7600	0.1
CDC STAR	0.02

It seems reasonable to expect that increases in computing speeds over the IBM 360/65 by factors of at least 16, 18, and 50, respectively, can be expected from the last three machines listed in Table 4.2. These last three machines can be obtained with 5×10^5 words or more of either fast core storage or slower extended core storage which, through clever programming, slows down computing speed only slightly. Thus, a program like 3DKIN could treat a problem with three neutron groups, one precursor group, and 5×10^4 or more spatial mesh points with all unknowns stored in fast or extended core storage, provided an excessive amount of geometrical detail were not specified for the problem.

Consider, then, the time which would be required on a machine which is 20 times faster than the IBM 360/65. Table 4.2 gives assurance that such machines are being built. A reasonable estimate for a problem with three neutron groups, one precursor group, and 5×10^4 mesh points on this machine would be

$$\text{time/step} = (1.1 \times 10^{-3})(.05)(5 \times 10^4)(3+.3) \text{ sec}$$
$$= 9.1 \text{ sec} .$$

Two hours of computing time would traverse about 800 time steps, enough to describe many interesting transients.

One goal set for the direct solution technique developed in this thesis has been that it provide benchmark solutions for difficult, practical problems. Solutions from the more rapid but more approximate synthesis techniques can then be compared against these. At the same time, the cost of obtaining these benchmark solutions must not be unduly great. The NSADE method appears to satisfy both of these criteria.

More importantly, the NSADE method is a practical method for the routine solution of several classes of problems, given that a very fast computer is available. One such class includes survey calculations where fine spatial detail is not required. Since more effort is required to prepare a problem for solution by a space-time synthesis method than for solution by the NSADE method, the synthesis methods lose much of their speed advantage when a number of different problems are to be run during a survey.

Space-time synthesis methods also have difficulty in treating problems where severe spatial flux tiltings and energy spectrum changes result. Selection of trial functions for such problems requires much insight and intuition. In contrast, the NSADE method requires only an initial flux distribution to start such a problem. Little insight is required as to how the solution will behave during the transient.

4.3 Limitations of the NSADE Method

The NSADE method is a more costly method than are space-time synthesis methods for a number of problems of interest to reactor designers. Once a reactor design has been finalized, there are a number of operating transients which need to be analyzed with fine spatial detail. Here, space-time synthesis methods are capable of providing sufficiently accurate solutions at a significantly lower cost.

Another factor may limit the effectiveness of the NSADE method on some current computing systems. Because this method tends to accumulate errors during the first few steps of a transient, a very accurate initial flux distribution and eigenvalue estimate must be used to start the calculations. All initial conditions used in this thesis were accurate to better than one part in 10^7 in the flux distribution and one part in 10^8 in the eigenvalue. Not only is it costly to obtain such an accurate initial condition, but it also is necessary to be able to carry 10 or more significant digits in all calculations. It would be difficult to utilize this method on any computing system which did not have floating-point capabilities which carry at least 10 significant decimal digits.

4.4 Recommendations for Further Work

The NSADE method can be easily extended to r-θ-z cylindrical geometry and to hexagonal-z geometry. Such extension would greatly increase the utility of the method in treating problems associated with several types of reactors.

It has been mentioned that it is possible to increase the time step size during certain parts of a transient, while it is necessary to decrease it during other parts if accuracy is to remain fairly constant throughout the transient. Algorithms which would automate this time step size variation should be investigated. It is probable that the rate of change of the frequencies, $\underline{\Omega}$, would provide an indication of when the time step size should be changed.

A final recommendation concerns the selection of the frequencies. There may well be algorithms which would select frequencies which would allow even larger time steps to be taken. This area of investigation deserves a great deal of attention.

REFERENCES

1. William H. Reed and K. F. Hansen, "Alternating Direction Methods for Reactor Kinetics Equations," Nucl. Sci. Eng. 41, 431 (1970).

2. William H. Reed, "Finite Difference Techniques for the Solution of the Reactor Kinetics Equations," Sc. D. Thesis, Department of Nuclear Engineering, Massachusetts Institute of Technology, MIT-NE-100 (May, 1969).

3. Richard S. Varga, Matrix Iterative Analysis, Chap. 6, Prentice-Hall, Englewood Cliffs, N. J. (1962).

4. A. F. Henry, "Space-Time Reactor Kinetics," 22.243 Course Notes, Massachusetts Institute of Technology (1969), unpublished.

5. S. Kaplan, O. J. Marlowe, and J. Bewick, "Application of Synthesis Techniques to Problems Involving Time Dependence," Nucl. Sci. Eng. 18, 163 (1964).

6. A. F. Henry, "The Application of Reactor Kinetics to the Analysis of Experiments," Nucl. Sci. Eng. 3, 52 (1958).

7. K. Ott, "Quasistatic Treatment of Spatial Phenomena in Reactor Dynamics," Nucl. Sci. Eng. 26, 563 (1966).

8. D. A. Meneley, K. Ott, and E. S. Wiener, "Space-Time Kinetics, the QX1 Code," ANL-7310, Argonne National Laboratory (1967).

9. J. B. Yasinsky and S. Kaplan, "Synthesis of Three-Dimensional Flux Shapes Using Discontinuous Sets of Trial Functions," Nucl. Sci. Eng. 28, 426 (1967).

10. J. B. Yasinsky, "Combined Space-Time Synthesis with Axially Discontinuous Trial Functions," WAPD-TM-736, Bettis Atomic Power Laboratory (1967).

11. J. B. Yasinsky, "The Solution of the Space-Time Neutron Group Diffusion Equations by a Time-Discontinuous Synthesis Method," Nucl. Sci. Eng. 29, 381 (1967).

12. E. L. Wachspress and M. Becker, "Variational Multichannel Synthesis with Discontinuous Trial Functions," KAPL-3095, Knolls Atomic Power Laboratory (1965).

13. W. M. Stacey, Jr., "A Variational Multichannel Space-Time Synthesis Method for Nonseparable Reactor Transients," Nucl. Sci. Eng. 34, 45 (1968).

14. K. F. Hansen and S. R. Johnson, "GAKIN, A Program for the Solution of the One-Dimensional, Multigroup, Space-Time Dependent Diffusion Equations," GA-7543, General Atomic (1967).

15. W. R. Cadwell, A. F. Henry, and A. J. Vigilotti, "WIGLE – A Program for the Solution of the Two-Group, Space-Time Diffusion Equations in Slab Geometry," WAPD-TM-416, Bettis Atomic Power Laboratory (1964).

16. J. B. Yasinsky, M. Natelson, and L. A. Hageman, "TWIGL – A Program to Solve the Two-Dimensional, Two-Group, Space-Time Neutron Diffusion Equations with Temperature Feedback," WAPD-TM-743, Bettis Atomic Power Laboratory (1968).

17. W. T. McCormick, Jr., "Numerical Solution of the Two-Dimensional Multigroup Kinetics Equations," Ph. D. Thesis, Department of Nuclear Engineering, Massachusetts Institute of Technology, MIT-NE-99 (May, 1969).

18. B. K. Larkin, "Some Stable Explicit Difference Approximations to the Diffusion Equation," Math. Comp. 18, 196 (1964).

19. S. J. Kast, "Solution of the Reactor Kinetics Equations in Two Dimensions by Finite Difference Methods," S. M. Thesis, Department of Nuclear Engineering, Massachusetts Institute of Technology (August, 1970).

20. Richard S. Varga, op. cit., Chap. 8.

21. D. W. Peaceman and H. H. Rachford, Jr., "The Numerical Solution of Parabolic and Elliptic Differential Equations," J. Soc. Ind. Appl. Math. 3, 28 (1955).

22. L. A. Hageman and J. B. Yasinsky, "Comparison of Alternating-Direction Time-Differencing Methods with Other Implicit Methods for the Solution of the Neutron Group-Diffusion Equations," Nucl. Sci. Eng. 38, 8 (1969).

23. A. L. Wight, K. F. Hansen, and D. R. Ferguson, "Application of Alternating-Direction Implicit Methods to the Space-Dependent Kinetics Equations," Nucl. Sci. Eng. 44, 239 (1971).

24. A. L. Wight, "The Application of Alternating-Direction Implicit Methods to the Space-Dependent Kinetics Equations," Ph. D. Thesis, Department of Nuclear Engineering, Massachusetts Institute of Technology (August, 1969).

25. Eugene L. Wachspress, Iterative Solution of Elliptic Systems, Chap. 3, Prentice-Hall, Englewood Cliffs, N. J. (1966).

26. Robert D. Richtmeyer and K. W. Morton, Difference Methods for Initial-Value Problems, Chap. 3, John Wiley and Sons, New York (1967).

27. L. A. Hageman, "Numerical Methods and Techniques Used in the Two-Dimensional Neutron-Diffusion Program PDQ-5," WAPD-TM-364, Bettis Atomic Power Laboratory (1963).

APPENDICES

Appendix A

THE SEMI-DISCRETE FORM OF THE SPACE-DEPENDENT REACTOR KINETICS EQUATIONS

The differential form of the space-dependent reactor kinetics equations has been given in Eqs. (1.1). These equations are repeated here for the sake of clarity.

$$\frac{1}{v_g}\frac{d\phi_g(\vec{r},t)}{dt} = \vec{\nabla}\cdot D_g(\vec{r},t)\,\vec{\nabla}\phi_g(\vec{r},t) + \sum_{g'=1}^{G}\Sigma_{gg'}(\vec{r},t)\,\phi_{g'}(\vec{r},t) + \sum_{i=1}^{I} f_{gi}C_i(\vec{r},t) \qquad (1 \leq g \leq G)$$

$$\frac{dC_i(\vec{r},t)}{dt} = -\lambda_i C_i(\vec{r},t) + \sum_{g'=1}^{G} p_{ig'}(\underline{r},t)\,\phi_{g'}(\vec{r},t) \qquad (1.1)$$

All of the symbols used here have been defined in section 1.2.

The discretization is carried out here in rectangular Cartesian coordinates. The region of interest is a rectangular parallelepiped. The origin of coordinates is placed in the lower front left corner of the parallelepiped, as shown in Fig. A.1.

The three-dimensional mesh is created by passing a series of planes, each of which is perpendicular to one of the three axes, entirely through the parallelepiped. The points of intersection of these planes, which lie within or on the boundaries of the parallelepiped, form the mesh. It is assumed that six of the planes are

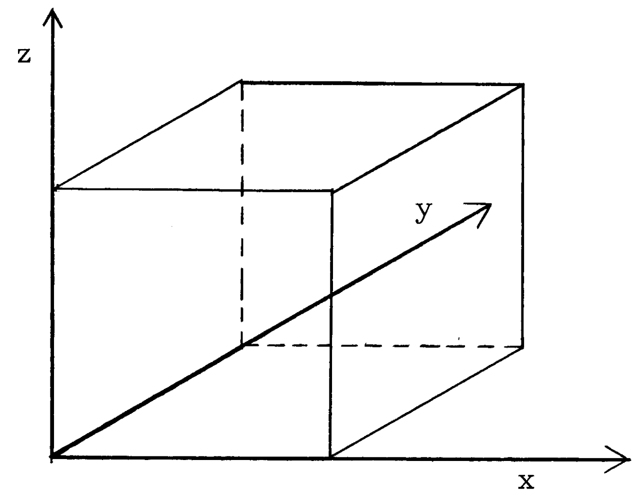

Fig. A.1. Coordinate System

coincident with the six faces so that planes of mesh points lie on the six faces. If a total of L, J, and K planes are passed perpendicular to the x-, y-, and z-axis, respectively, there are a total of L×J×K points in the mesh within or on the boundaries of the parallelepiped.

Figures A.2a and A.2b depict, respectively, planes perpendicular to the z-axis and y-axis which pass through mesh point (l, j, k).

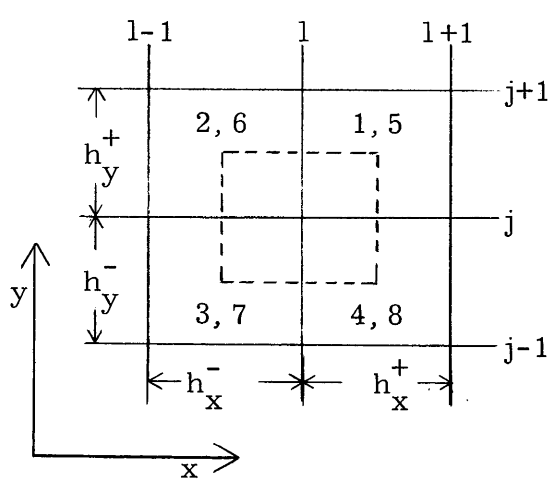

Fig. A.2a. Plane Perpendicular to z-Axis at (l, j, k)

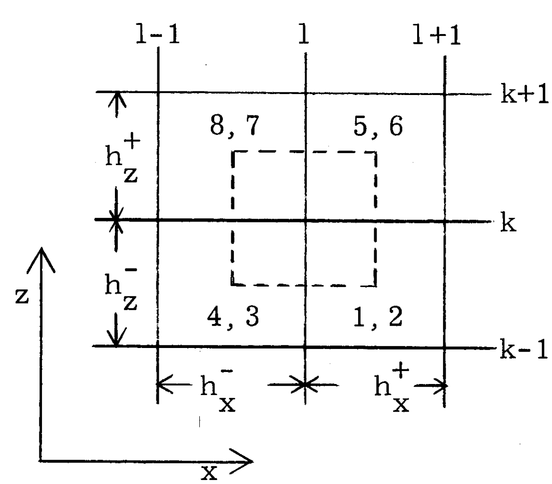

Fig. A.2b. Plane Perpendicular to y-Axis at (l, j, k)

The broken lines lie exactly halfway between the solid mesh lines. The eight octants which touch on point (l, j, k) are numbered as shown above. Octants 1, 2, 3, and 4 lie below the z-plane passing through point (l, j, k), while octants 5, 6, 7, and 8 lie above it.

The discrete equations for point (i, j, k) are obtained by integrating Eqs. (1.1) over the volume contained within $x_l - h_x^-/2 \leq x \leq x_l + h_x^+/2$, $y_j - h_y^-/2 \leq y \leq y_j + h_y^+/2$, and $z_k - h_z^-/2 \leq z \leq z_k + h_z^+/2$. It is assumed that the material within each octant is homogeneous. In the derivation that follows, superscripts on material constants denote the octants in which the materials lie.

$$\frac{1}{v_g} \frac{d}{dt} \int_{z_k-h_z^-/2}^{z_k+h_z^+/2} dz \int_{y_j-h_y^-/2}^{y_j+h_y^+/2} dy \int_{x_l-h_x^-/2}^{x_l+h_x^+/2} dx\, \phi_g(x, y, z, t) =$$

$$\int_{z_k-h_z^-/2}^{z_k+h_z^+/2} dz \int_{y_j-h_y^-/2}^{y_j+h_y^+/2} dy \int_{x_l-h_x^-/2}^{x_l+h_x^+/2} dx \times$$

$$\left\{ \vec{\nabla} \cdot D_g(x, y, z, t) \vec{\nabla} \phi_g(x, y, z, t) + \sum_{g'=1}^{G} \Sigma_{gg'}(x, y, z, t)\, \phi_{g'}(x, y, z, t) + \right.$$

$$\left. \sum_{i=1}^{I} f_{gi} C_i(x, y, z, t) \right\}, \qquad (1 \leq g \leq G). \tag{A.1}$$

With the following definitions,

$$\phi_{g,l,j,k} = \frac{1}{V_{l,j,k}} \int \int \int \phi_g(x, y, z)\, dx\, dy\, dz \tag{A.2a}$$

$$C_{i,l,j,k} = \frac{1}{V_{l,j,k}} \int \int \int C_i(x, y, z)\, dx\, dy\, dz \tag{A.2b}$$

$$V_{l,j,k} = \frac{1}{8}\left(h_x^+ + h_x^-\right)\left(h_y^+ + h_y^-\right)\left(h_z^+ + h_z^-\right) \tag{A.2c}$$

and

$$\Sigma_{gg',l,j,k} = \frac{1}{8}\Big[h_x^+ h_y^+ h_z^- \Sigma_{gg'}^1 + h_x^- h_y^+ h_z^- \Sigma_{gg'}^2 + h_x^- h_y^- h_z^- \Sigma_{gg'}^3 + h_x^+ h_y^- h_z^- \Sigma_{gg'}^4 + h_x^+ h_y^+ h_z^+ \Sigma_{gg'}^5 + h_x^- h_y^+ h_z^+ \Sigma_{gg'}^6 + h_x^- h_y^- h_z^+ \Sigma_{gg'}^7 + h_x^+ h_y^- h_z^+ \Sigma_{gg'}^8\Big], \tag{A.2d}$$

where the integrals are taken over the limits shown in Eq. (A.1), Eq. (A.1) becomes

$$\frac{V_{l,j,k}}{v_g}\frac{d\phi_{g,l,j,k}}{dt} = \sum_{m=1}^{6}\left\{\int d\vec{s}_m \cdot D_g(x,y,z)\,\vec{\nabla}\phi_g(x,y,z)\right\} + \sum_{g'=1}^{G} \Sigma_{gg',l,j,k}\,\phi_{g',l,j,k} + V_{l,j,k}\sum_{i=1}^{I} f_{gi}\,C_{i,l,j,k}\,. \tag{A.3}$$

In Eq. (A.3), the volume integral for the diffusion terms has been changed to a surface integral, using Gauss' theorem. The summation over m indicates that the integral has been broken into integrals over the six faces of the volume. For illustrative purposes, consider the face which is perpendicular to the x-axis at $x = x_l + h_x^+/2$. The surface integral for this face is given by

$$\int_{z-h_z^-/2}^{z+h_z^+/2} dz \int_{y-h_y^-/2}^{y+h_y^+/2} dy \left\{D_g(x,y,z)\,\vec{\nabla}\phi_g(x,y,z)\cdot\vec{n}_x\right\}\Big|_{x=x_l+h_x^+/2},$$

where $\vec{n}_x$ is a unit vector in the positive x-direction.

In order to carry out this integration, the current normal to the face is approximated by a simple finite difference:

$$\vec{\nabla}\phi_g(x,y,z)\cdot\vec{n}_x \doteq \frac{\phi_{g,l+1,j,k} - \phi_{g,l,j,k}}{h_x^+} \quad . \tag{A.4}$$

With this approximation, the surface integral representing leakage across the face at $x=x_l+h_x^+/2$ becomes

$$\int_{z-h_z^-/2}^{z+h_z^+/2} dz \int_{y-h_y^-/2}^{y+h_y^+/2} dy \left\{ D_g(x,y,z)\,\vec{\nabla}\phi_g(x,y,z)\cdot\vec{n}_x \right\}\Bigg|_{x=x_l+h_x^+/2} \doteq$$

$$\left(\frac{\phi_{g,l+1,j,k}-\phi_{g,l,j,k}}{h_x^+}\right)\cdot\left(\frac{D_g^1h_y^+h_z^-}{4}+\frac{D_g^5h_y^+h_z^+}{4}+\frac{D_g^8h_y^-h_z^+}{4}+\frac{D_g^4h_y^-h_z^-}{4}\right)$$

$$= R_{g,l+\frac{1}{2},j,k}(\phi_{g,l+1,j,k}-\phi_{g,l,j,k}) , \tag{A.5}$$

where $R_{g,l+\frac{1}{2},j,k}$ has been defined as

$$R_{g,l+\frac{1}{2},j,k} = \frac{1}{4h_x^+}\left(D_g^1h_y^+h_z^-+D_g^5h_y^+h_z^++D_g^8h_y^-h_z^++D_g^4h_y^-h_z^-\right) . \tag{A.6a}$$

By defining five more leakage coefficients as

$$R_{g,l-\frac{1}{2},j,k} = \frac{1}{4h_x^-}\left(D_g^2h_y^+h_z^-+D_g^6h_y^+h_z^++D_g^7h_y^-h_z^++D_g^3h_z^-h_y^-\right) , \tag{A.6b}$$

$$R_{g,l,j+\frac{1}{2},k} = \frac{1}{4h_y^+}\left(D_g^1h_x^+h_z^-+D_g^5h_x^+h_z^++D_g^6h_x^-h_z^++D_g^2h_x^-h_z^-\right) , \tag{A.6c}$$

$$R_{g,l,j-\frac{1}{2},k} = \frac{1}{4h_y^-}\left(D_g^4h_x^+h_z^-+D_g^8h_x^+h_z^++D_g^7h_x^-h_z^++D_g^3h_x^-h_z^-\right) , \tag{A.6d}$$

$$R_{g,l,j,k+\frac{1}{2}} = \frac{1}{4h_z^+}\left(D_g^8h_x^+h_y^-+D_g^7h_x^-h_y^-+D_g^6h_x^-h_y^++D_g^5h_x^+h_y^+\right) , \tag{A.6e}$$

$$R_{g,l,j,k-\frac{1}{2}} = \frac{1}{4h_z^-}\left(D_g^4h_x^+h_y^-+D_g^3h_x^-h_y^-+D_g^2h_x^-h_y^++D_g^1h_x^+h_y^+\right) , \tag{A.6f}$$

Eq. (A.3) can be written in its final form as

$$\frac{d\phi_{g,l,j,k}}{dt} = v_g \Bigg\{ \frac{1}{V_{l,j,k}} \Big[R_{g,l+\frac{1}{2},j,k}(\phi_{g,l+1,j,k} - \phi_{g,l,j,k}) + R_{g,l-\frac{1}{2},j,k}(\phi_{g,l-1,j,k} - \phi_{g,l,j,k}) + R_{g,l,j+\frac{1}{2},k}(\phi_{g,l,j+1,k} - \phi_{g,l,j,k}) + R_{g,l,j-\frac{1}{2},k}(\phi_{g,l,j-1,k} - \phi_{g,l,j,k}) + R_{g,l,j,k+\frac{1}{2}}(\phi_{g,l,j,k+1} - \phi_{g,l,j,k}) + R_{g,l,j,k-\frac{1}{2}}(\phi_{g,l,j,k-1} - \phi_{g,l,j,k}) + \sum_{g'=1}^{G} \Sigma_{gg',l,j,k}\,\phi_{g',l,j,k} \Big] + \sum_{i=1}^{I} f_{gi} C_{i,l,j,k} \Bigg\}, \quad (1 \leqslant g \leqslant G). \qquad \text{(A.7)}$$

Furthermore, by defining the (LJK) $\times$ (LJK) square matrices

$$\underline{T}_{gg'} = \text{diag}\{v_g \Sigma_{gg',l,j,k}/V_{l,j,k}\}, \qquad \text{(A.8a)}$$

$$\underline{F}_{gi} = \text{diag}\{v_g f_{gi}\}, \qquad \text{(A.8b)}$$

and $\underline{D}_g$ such that

$$\underline{D}_g \vec{\psi}_g = v_g \text{ col} \Bigg\{ \frac{1}{V_{l,j,k}} \Big[R_{g,l+\frac{1}{2},j,k}(\phi_{g,l+1,j,k} - \phi_{g,l,j,k}) + R_{g,l-\frac{1}{2},j,k}(\phi_{g,l-1,j,k} - \phi_{g,l,j,k}) + R_{g,l,j+\frac{1}{2},k}(\phi_{g,l,j+1,k} - \phi_{g,l,j,k}) + R_{g,l,j-\frac{1}{2},k}(\phi_{g,l,j-1,k} - \phi_{g,l,j,k}) + R_{g,l,j,k+\frac{1}{2}}(\phi_{g,l,j,k+1} - \phi_{g,l,j,k}) + R_{g,l,j,k-\frac{1}{2}}(\phi_{g,l,j,k-1} - \phi_{g,l,j,k}) \Big] \Bigg\}, \qquad \text{(A.8c)}$$

the equations for all mesh points can be combined into the single matrix equation

$$\frac{d\vec{\psi}_g}{dt} = \underline{D}_g \vec{\psi}_g + \sum_{g'=1}^{G} \underline{T}_{gg'} \vec{\psi}_{g'} + \sum_{i=1}^{I} \underline{F}_{gi} \vec{C}_i , \qquad (1 \leq g \leq G) . \qquad (1.4)$$

Here, the vectors $\vec{\psi}_g$ and $\vec{C}_i$ are formed by ordering the group g fluxes and delayed precursor group i concentrations, respectively, in a consistent manner.

The discrete equation for the i^{th} delayed precursor concentration at point (l, j, k) is derived in an analogous fashion. It is given by

$$\frac{dC_{i,l,j,k}}{dt} = -\lambda_i C_{i,l,j,k} + \frac{1}{V_{l,j,k}} \sum_{g'=1}^{G} P_{ig',l,j,k} \phi_{g',l,j,k} , \qquad (1 \leq i \leq I) , \qquad (A.9)$$

where

$$P_{ig',l,j,k} = \frac{\beta_i}{8} \Big[h_x^+ h_y^+ h_z^- \nu_{g'}^1 \Sigma_{fg'}^1 + h_x^- h_y^+ h_z^- \nu_{g'}^2 \Sigma_{fg'}^2 + h_x^- h_y^- h_z^- \nu_{g'}^3 \Sigma_{fg'}^3 + h_x^+ h_y^- h_z^- \nu_{g'}^4 \Sigma_{fg'}^4 + h_x^+ h_y^+ h_z^+ \nu_{g'}^5 \Sigma_{fg'}^5 + h_x^- h_y^+ h_z^+ \nu_{g'}^6 \Sigma_{fg'}^6 + h_x^- h_y^- h_z^+ \nu_{g'}^7 \Sigma_{fg'}^7 + h_x^+ h_y^- h_z^+ \nu_{g'}^8 \Sigma_{fg'}^8 \Big] . \qquad (A.10)$$

By defining the (LJK) by (LJK) matrices

$$\underline{\Lambda}_i = \lambda_i \underline{I} \qquad (A.11a)$$

and

$$\underline{P}_{ig'} = \mathrm{diag}\left\{ P_{ig',l,j,k} / V_{l,j,k} \right\} , \qquad (A.11b)$$

Eq. (A.9) for all mesh points can be written in matrix form as

$$\frac{d\vec{C}_i}{dt} = -\underline{\Lambda}_i \vec{C}_i + \sum_{g'=1}^{G} \underline{P}_{ig'} \vec{\psi}_{g'} . \qquad (1.5)$$

Appendix B

THEOREMS

Several theorems and lemmas were offered without proof in Chapter 2. They are restated and proved here.

LEMMA 1.[1] The operators $\underline{C}_1(\underline{\Omega}, h)$ and $\underline{C}_2(\underline{\Omega}, h)$ are consistent.

Proof. The consistency condition requires that

$$\left\| \frac{\vec{\theta}(t+h) - \underline{C}_1(\underline{\Omega}, h)\vec{\theta}(t)}{h} \right\| \to 0 \text{ as } h \to 0 . \tag{B.1}$$

An identical condition must hold for $\underline{C}_2(\underline{\Omega}, h)$. Only $\underline{C}_1(\underline{\Omega}, h)$ will be treated here. The proof for $\underline{C}_2(\underline{\Omega}, h)$ is identical.

The numerator in Eq. (B.1) can be written in the form

$$\vec{\theta}(t+h) - \underline{C}_1\vec{\theta}(t) = e^{\underline{\Omega}h}[\underline{I} - h(\underline{D}_1+\underline{E}_4-\alpha\underline{\Omega})]^{-1}$$

$$\times \{[\underline{I} - h(\underline{D}_1+\underline{E}_4 - \alpha\underline{\Omega})]e^{-\underline{\Omega}h}\vec{\theta}(t+h)$$

$$- [\underline{I}+h(\underline{D}_2+\underline{E}_3 - \gamma\underline{\Omega})]\}\vec{\theta}(t) .$$

Expanding $e^{\underline{\Omega}h}$ and $\vec{\theta}(t+h)$ in a Taylor's series gives

$$\vec{\theta}(t+h) - \underline{C}_1\vec{\theta}(t) = e^{\underline{\Omega}h}[\underline{I} - h(\underline{D}_1+\underline{E}_4 - \alpha\underline{\Omega})]^{-1}$$

$$\times \{h\frac{d\vec{\theta}(t)}{dt} - h\underline{A}\vec{\theta}(t) + O(h^2)\} .$$

It has been stated in section 2.1 that

$$\underline{M}\vec{\theta}(t) = \underline{A}\vec{\theta}(t) + O(\Delta x^2) + O(\Delta y^2) + O(\Delta z^2) .$$

Therefore,

$$\left\| \frac{\vec{\theta}(t+h) - \underline{C}_1 \vec{\theta}(t)}{h} \right\| = \left\| e^{\underline{\Omega} h} [\underline{I} - h(\underline{D}_1 + \underline{E}_4 - \alpha \underline{\Omega})]^{-1} \right\| \times \{ O(h) + O(\Delta x^2) + O(\Delta y^2) + O(\Delta z^2) \},$$

$$\left\| \frac{\vec{\theta}(t+h) - \underline{C}_1 \vec{\theta}(t)}{h} \right\| \leq \left\| e^{\underline{\Omega} h} [\underline{I} - h(\underline{D}_1 + \underline{E}_4 - \alpha \underline{\Omega})]^{-1} \right\| \times \| O(h) + O(\Delta x^2) + O(\Delta y^2) + O(\Delta z^2) \| . \tag{B.2}$$

Theorem 3, proved later in this Appendix gives assurance that $\| e^{\underline{\Omega} h} [\underline{I} - h(\underline{D}_1 + \underline{E}_4 - \alpha \underline{\Omega})]^{-1} \|$ is bounded for the L_2 norm provided the ratios $h/\Delta x^2$, $h/\Delta y^2$, and $h/\Delta z^2$ are fixed, real constants of any finite size. Calling this bound K allows Eq. (B.2) to be written as

$$\left\| \frac{\vec{\theta}(t+h) - \underline{C}_1(\underline{\Omega}, h) \vec{\theta}(t)}{h} \right\| \leq \underline{K} \| O(h) \|$$

for the L_2 norm. Thus, $\underline{C}_1(\underline{\Omega}, h)$ satisfies the consistency condition.

LEMMA 2.[1] If two operators are consistent, then their product is consistent.

<u>Proof</u>. Let $\underline{C}_1$ and $\underline{C}_2$ be two consistent operators, i.e.,

$$\left\| \frac{\vec{\theta}(t+h) - \underline{C}_2 \vec{\theta}(t)}{h} \right\| \rightarrow 0 \text{ as } h \rightarrow 0$$

$$\left\| \frac{\vec{\theta}(t+2h) - \underline{C}_1 \vec{\theta}(t+h)}{h} \right\| \rightarrow 0 \text{ as } h \rightarrow 0 .$$

Since $\underline{C}_1$ is consistent, it has a bounded norm so that

$$\|\underline{C}_1\| \left\| \frac{\vec{\theta}(t+h) - \underline{C}_2\vec{\theta}(t)}{h} \right\| \to 0 \text{ as } h \to 0 .$$

The definition of a norm provides that $\|\underline{C}\vec{x}\| \leqslant \|\underline{C}\| \|\vec{x}\|$. Therefore,

$$\|\underline{C}_1\vec{\theta}(t+h) - \underline{C}_1\underline{C}_2\vec{\theta}(t)\| \to 0 \text{ as } h \to 0 .$$

Using the triangle inequality, $\|\vec{x} + \vec{y}\| \leqslant \|\vec{x}\| + \|\vec{y}\|$, this becomes

$$\left\| \frac{\vec{\theta}(t+2h) - \underline{C}_1\vec{\theta}(t+h)}{h} + \frac{\underline{C}_1\vec{\theta}(t+h) - \underline{C}_1\underline{C}_2\vec{\theta}(t)}{h} \right\| \to 0 \text{ as } h \to 0$$

or

$$\left\| \frac{\vec{\theta}(t+2h) - \underline{C}_1\underline{C}_2\vec{\theta}(t)}{h} \right\| \to 0 \text{ as } h \to 0, \tag{B.3}$$

which is the consistency requirement for the product.

THEOREM 3.[1] A family of matrices $\underline{M}_n$ of varying dimension n having at most $\ell < n$ nonzero elements in each row or column, ℓ being constant for all n, has a uniform L_2 bound if the individual elements of the matrices $\underline{M}_n$ are uniformly bounded for all n.

Proof. Let $c > 0$ be a bound on the absolute value of the individual elements, $m^n_{j,k}$, of the matrices $\underline{M}_n$. Then

$$\max_k \sum_{j=1}^{n} |m^n_{j,k}| \leqslant c\ell$$

$$\max_j \sum_{k=1}^{n} |m^n_{j,k}| \leqslant c\ell$$

for all n. However, by definition,

$$\|\underline{M}_n\|_2^2 = \sup_{\|\vec{x}\|_2=1} \sum_{j=1}^{n} \left| \sum_{k=1}^{n} m_{j,k}^n x_k \right|^2 .$$

The Cauchy-Schwarz inequality gives

$$\|\underline{M}_n\|_2^2 \leq \sup_{\|\vec{x}\|_2=1} \sum_{j=1}^{n} \left\{ \left[\sum_{k=1}^{n} |m_{j,k}^n| \right] \left[\sum_{k=1}^{n} |m_{j,k}^n x_k^2| \right] \right\}$$

$$\leq \sup_{\|\vec{x}\|_2=1} \sum_{j=1}^{n} \left\{ c\ell \sum_{k=1}^{n} |m_{j,k}^n x_k^2| \right\}$$

$$\leq \sup_{\|\vec{x}\|_2=1} c\ell \sum_{k=1}^{n} \left\{ |x_k|^2 \sum_{j=1}^{n} |m_{j,k}^n| \right\}$$

$$\leq (c\ell)^2 \sup_{\|\vec{x}\|_2=1} \sum_{k=1}^{n} |x_k|^2 ,$$

$$\|\underline{M}_n\|_2^2 \leq (c\ell)^2 ,$$

or

$$\|\underline{M}_n\| \leq c\ell , \tag{B.4}$$

and the theorem is proved.

THEOREM 4.[1] The matrices $(\underline{I} - h\underline{R})^{-1}$ and $(\underline{I} + hR)(\underline{I} - h\underline{R})^{-1}$ have L_2 norms of less than unity provided that $(\underline{R} + \underline{R}^T)$ is negative definite.

Proof. By definition,

$$\|(\underline{I} - h\underline{R})^{-1}\|_2^2 = \max_{\vec{v}} \frac{\vec{v}^T(\underline{I} - h\underline{R}^T)^{-1}(\underline{I} - h\underline{R})^{-1}\vec{v}}{\vec{v}^T\vec{v}} .$$

Let $\vec{u} = (\underline{I} - h\underline{R})^{-1}\vec{v}$. Then

$$\left\|(\underline{I} - h\underline{R})^{-1}\right\|_2^2 = \max_{\vec{u}} \frac{\vec{u}^T\vec{u}}{\vec{u}^T(\underline{I} - h\underline{R}^T)(\underline{I} - h\underline{R})\vec{u}}$$

$$= \max_{\vec{u}} \frac{\vec{u}^T\vec{u}}{\vec{u}^T[\underline{I} - h(\underline{R}^T + \underline{R}) + h^2\underline{R}^T\underline{R}]\vec{u}} . \qquad \text{(B.5)}$$

If $(\underline{R}^T + \underline{R})$ is negative definite, the denominator of Eq. (B.5) is positive and larger than the numerator. Therefore,

$$\left\|(\underline{I} - h\underline{R})^{-1}\right\|_2 < 1 .$$

Likewise, for the product $(\underline{I} + h\underline{R})(\underline{I} - h\underline{R})^{-1}$, the L_2 norm is defined as

$$\left\|(\underline{I} + h\underline{R})(\underline{I} - h\underline{R})^{-1}\right\|_2^2 = \max_{\vec{v}} \frac{\vec{v}^T(\underline{I} - h\underline{R}^T)^{-1}(\underline{I} + h\underline{R}^T)(\underline{I} + h\underline{R})(\underline{I} - h\underline{R})^{-1}\vec{v}}{\vec{v}^T\vec{v}}$$

With $\vec{u}$ defined as before, this becomes

$$\left\|(\underline{I} + h\underline{R})(\underline{I} - h\underline{R})^{-1}\right\|_2^2 = \max_{\vec{u}} \frac{\vec{u}^T(\underline{I} + h\underline{R}^T)(\underline{I} + h\underline{R})\vec{u}}{\vec{u}^T(\underline{I} - h\underline{R}^T)(\underline{I} - h\underline{R})\vec{u}}$$

$$= \max_{\vec{u}} \frac{\vec{u}^T[\underline{I} + h(\underline{R}^T + \underline{R}) + h^2\underline{R}^T\underline{R}]\vec{u}}{\vec{u}^T[\underline{I} - h(\underline{R}^T + \underline{R}) + h^2\underline{R}^T\underline{R}]\vec{u}} . \qquad \text{(B.6)}$$

Again, if $(\underline{R}^T + \underline{R})$ is negative definite, the denominator of Eq. (B.6) is larger than the numerator so that

$$\left\|(\underline{I} + h\underline{R})(\underline{I} - h\underline{R})^{-1}\right\|_2 < 1 .$$

THEOREM 5.[24] As t approaches infinity, the solution vector $\vec{\psi}(t) = e^{\underline{A}t}\vec{\psi}_o$ approaches $\alpha e^{\omega_o t}\vec{e}_o$, where ω_o is the largest eigenvalue of $\underline{A}$, $\vec{e}_o$ the corresponding eigenvector, and $\alpha = (\vec{\psi}_o, \vec{e}_o)$.

Proof. Write $\vec{\psi}_o$ as a linear combination of $\vec{e}_o$ and $\vec{v}$, where $(\vec{v}, \vec{e}_o) = 0$, that is, $\vec{\psi}_o = \alpha\vec{e}_o + \beta\vec{v}$. Now,

$$\alpha(\vec{e}_o, \vec{e}_o) + \beta(\vec{e}_o, \vec{v}) = (\vec{e}_o, \vec{\psi}_o)$$

or

$$\alpha = (\vec{e}_o, \vec{\psi}_o),$$

if $(\vec{e}_o, \vec{e}_o)$ is normalized to unity.

Write $\vec{\psi}(t)$ as

$$\begin{aligned}\vec{\psi}(t) &= e^{\underline{A}t}(\alpha\vec{e}_o + \beta\vec{v}) \\ &= \alpha e^{\omega_o t}\vec{e}_o + \beta e^{\underline{A}t}\vec{v} \\ &= \alpha e^{\omega_o t}\left[\vec{e}_o + (\beta/\alpha)\, e^{\underline{B}t}\vec{v}\right],\end{aligned} \tag{B.7}$$

where

$$\underline{B} = \underline{A} - \omega_o\underline{I}.$$

Note that the largest eigenvalue of $\underline{B}$ is 0, and all the others are given by $\lambda_i = \omega_i - \omega_o$ and have real parts less than zero.

Now, put $\underline{B}$ in Jordan form:

$$\underline{J} = \underline{S}^{-1}\underline{B}\underline{S} = \begin{bmatrix} \underline{J}_1 & & & \underline{0} \\ & \underline{J}_2 & & \\ & & \underline{J}_3 & \\ \underline{0} & & & \ddots \end{bmatrix}, \tag{B.8}$$

where each of the blocks on the diagonal is of the form

$$\underline{J}_i = \begin{bmatrix} \lambda_i & 1 & & 0 \\ & \lambda_i & 1 & \\ & & \lambda_i & 1 \\ 0 & & & \ddots \end{bmatrix} . \tag{B.9}$$

$\underline{J}_i$ is a p_i by p_i matrix, p_i being less than or equal to the multiplicity of the i^{th} eigenvalue, and the λ_i's are arranged in order of non-increasing real part. $\underline{J}_1$ is a 1X1 matrix since the largest eigenvalue of $\underline{B}$ is simple.

Now

$$e^{\underline{B}t}\vec{v} = e^{\underline{S}^{-1}\underline{J}\underline{S}t}\vec{v}$$

$$= (\underline{I} + \underline{S}^{-1}(\underline{J}t)\underline{S} + (1/2!)\underline{S}^{-1}(\underline{J}t)^2\underline{S} + \dots)\vec{v}$$

$$= \underline{S}^{-1} e^{\underline{J}t} \underline{S}\vec{v} = \underline{S}^{-1} e^{\underline{J}t} \underline{a} , \tag{B.10}$$

where $\vec{a} = \underline{S}\vec{v}$. But

$$e^{\underline{J}t} = \begin{bmatrix} 1 & & & \underline{0} \\ & e^{\underline{J}_2 t} & & \\ & & e^{\underline{J}_3 t} & \\ \underline{0} & & & \ddots \end{bmatrix} . \tag{B.11}$$

Since $\underline{A}$ and $\underline{B}$ share the same eigenvectors, $\vec{e}_o$ is the eigenvector of $\underline{B}$ corresponding to eigenvalue 0, and the transformation $\underline{S}$ also puts $\underline{A}$ into Jordan form. That is,

$$\underline{J}'\underline{S} = \underline{S}\underline{A}, \quad \underline{S}^{-1}\underline{J}'\underline{S}\vec{e}_o = \underline{A}\vec{e}_o = \omega_o\vec{e}_o, \quad \underline{J}'\underline{S}\vec{e}_o = \omega_o\underline{S}\vec{e}_o ,$$

where

$$\underline{J}' = \begin{bmatrix} \omega_o & & & \underline{0} \\ & \underline{J}'_2 & & \\ & & \underline{J}'_3 & \\ \underline{0} & & & \ddots \end{bmatrix} . \tag{B.12}$$

Thus

$$\underline{S}\vec{e}_o = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \quad \text{and} \quad \vec{S} = \begin{bmatrix} \vec{e}_o^{\,T} \\ x \\ x \\ \\ \cdots \end{bmatrix} ,$$

so that

$$\underline{S}\vec{v} = \begin{bmatrix} \vec{e}_o^{\,T}\vec{v} \\ x \\ x \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ x \\ x \\ \vdots \end{bmatrix} .$$

The first element of $\underline{S}\vec{v}$ is zero since $\vec{e}_o$ is orthogonal to $\vec{v}$.

Now

$$e^{\underline{J}t}\underline{S}\vec{v} = \begin{bmatrix} 1 & & & \\ & e^{\underline{J}_2 t} & & \underline{0} \\ & & e^{\underline{J}_3 t} & \\ \underline{0} & & & \ddots \end{bmatrix} \begin{bmatrix} 0 \\ \vec{a}_2 \\ \vec{a}_3 \\ \vdots \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ e^{\underline{J}_2 t}\vec{a}_2 \\ e^{\underline{J}_3 t}\vec{a}_3 \\ \vdots \end{bmatrix} . \qquad \text{(B.13)}$$

Hence, $\|\underline{S}^{-1} e^{\underline{J}t}\underline{S}\vec{v}\| \leq \|\underline{S}^{-1}\| \sum_{i=2}^{n} \|e^{\underline{J}_i t}\| \cdot \|\vec{a}_i\|$, which approaches

$$\|\underline{S}^{-1}\| \sum_{i=2}^{n} \|\vec{a}_i\| \frac{t^{p_i - 1}}{(p_i - 1)} e^{t \cdot Re(\lambda_i)}$$

as t approaches infinity, using Lemma 8.1 from Ref. 20. Since $Re(\lambda_i)$ is less than zero, all $i > 1$, this norm goes to zero for large t. Hence, $\|(\beta/\alpha)\, e^{\underline{B}t}\vec{v}\|$ approaches zero as t approaches infinity, and the vector $\vec{e}_o + (\beta/\alpha)\, e^{\underline{B}t}\vec{v}$ approaches $\vec{e}_o$, completing the proof.

THEOREM 6.[2] If $\underline{\Omega} = \omega_o I$, the approximate solution operator $\underline{B}(\underline{\Omega}, h)$ has as its largest eigenvalue $e^{2\omega_o h}$ with corresponding eigenvalue $\vec{e}_o$, where $\underline{A}\vec{e}_o = \omega_o \vec{e}_o$.

Proof. Letting $\underline{\Omega} = \omega_o I$,

$$\underline{B}(\omega_o \underline{I}, h)\vec{e}_o = e^{\omega_o h}[\underline{I} - h(\underline{A}_4 - \alpha\omega_o \underline{I})]^{-1}[\underline{I} + h(\underline{A}_3 - \alpha\omega_o \underline{I})]$$

$$\times [\underline{I} - h(\underline{A}_2 - \alpha\omega_o \underline{I})]^{-1}[\underline{I} + h(\underline{A}_1 - \gamma\omega_o \underline{I})] e^{\omega_o h} \vec{e}_o .$$

But

$$[\underline{I} + h(\underline{A}_3 - \gamma\omega_o \underline{I})]\vec{e}_o = [\underline{I} - h(\underline{A}_4 - \alpha\omega_o \underline{I})]\vec{e}_o$$

and

$$[\underline{I} + h(\underline{A}_1 - \gamma\omega_o \underline{I})]\vec{e}_o = [\underline{I} - h(\underline{A}_2 - \alpha\omega_o \underline{I})]\vec{e}_o .$$

Therefore,

$$\underline{B}(\omega_o \underline{I}, h)\vec{e}_o = e^{\omega_o h}[\underline{I} - h(\underline{A}_4 - \alpha\omega_o \underline{I})]^{-1}[\underline{I} + h(\underline{A}_3 - \gamma\omega_o \underline{I})]$$

$$\times [\underline{I} - h(\underline{A}_2 - \alpha\omega_o \underline{I})]^{-1}[\underline{I} - h(\underline{A}_2 - \alpha\omega_o \underline{I})] e^{\omega_o h} \vec{e}_o$$

$$= e^{\omega_o h}[\underline{I} - h(\underline{A}_4 - \alpha\omega_o \underline{I})]^{-1}[\underline{I} - h(\underline{A}_4 - \alpha\omega_o \underline{I})] e^{\omega_o h} \vec{e}_o$$

or

$$\underline{B}(\omega_o \underline{I}, h)\vec{e}_o = e^{2\omega_o h} \vec{e}_o . \tag{B.14}$$

Appendix C

TEST PROBLEM DATA

The reactor parameters for the four configurations used in Chapter 3 for three-dimensional experiments are presented in this appendix. The symbols used in this appendix are defined as follows:

Δx = mesh spacing (cm) in x-direction

Δy = mesh spacing (cm) in y-direction

Δz = mesh spacing (cm) in z-direction

λ_i = decay constant (sec^{-1}) of i^{th} precursor

β_i = delay fraction of i^{th} precursor

χ_{gi} = fraction of decays of i^{th} precursor which yield neutrons in group g

v_g = velocity of g^{th} neutron group (cm/sec)

χ_g = prompt fission spectrum component for group g

Σ_{tr} = macroscopic transport cross section (cm^{-1})

$D = 1/(3\Sigma_{tr})$ = diffusion coefficient (cm)

Σ_a = macroscopic absorption cross section (cm^{-1})

Σ_f = macroscopic fission cross section (cm^{-1})

ν = average number of neutrons per fission

$\Sigma_{J \to J+1}$ = macroscopic scattering cross section from group J to group J+1 (cm^{-1}).

Unless otherwise noted, all boundary conditions are homogeneous Dirichlet.

Configuration 1

Number of neutron groups = 2

Number of precursor groups = 1

Geometry: Homogeneous cube, 200 cm on a side

$\Delta x = \Delta y = \Delta z = 20$ cm

Precursor Constants:

$\lambda_i = .08$, $\beta_i = .0064$, $\chi_{11} = 1.0$, $\chi_{21} = 0.0$

Material Properties:

	Group 1	Group 2
v	3.0×10^7	2.2×10^5
χ	1.0	0.0
Σ_{tr}	.2468	.3084
Σ_a	.001382	.0054869
ν	2.41	2.41
Σ_f	.000242	.00408
$\Sigma_{J \to J+1}$	.0023	0.0

Initial Conditions:

Spatial shape: cosine

Critical k_{eff}: .895285417

Configuration 2

Number of neutron groups = 2

Number of precursor groups = 1

Geometry:

$$\Delta x = \Delta y = \Delta z = 8.0 \text{ cm}$$

$$\text{height} = 160 \text{ cm (z-direction)}$$

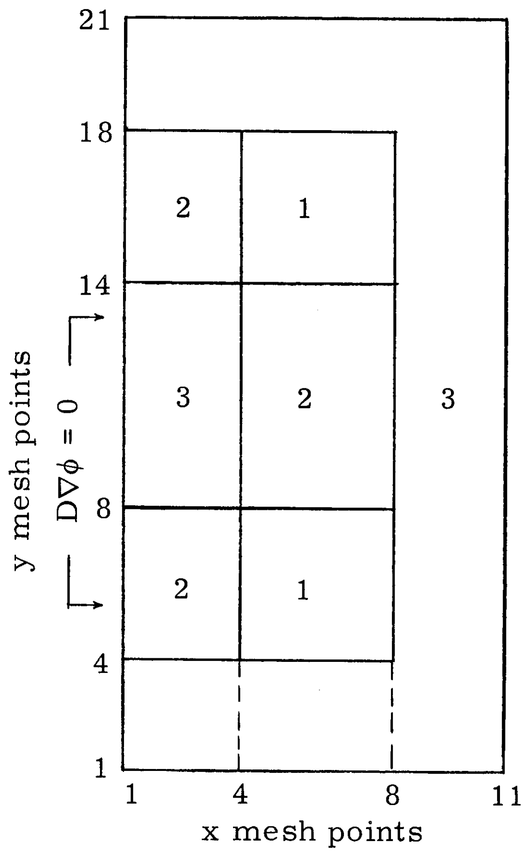

Fig. C.1a. x-y Plane for $24 \leq z \leq 136$ cm

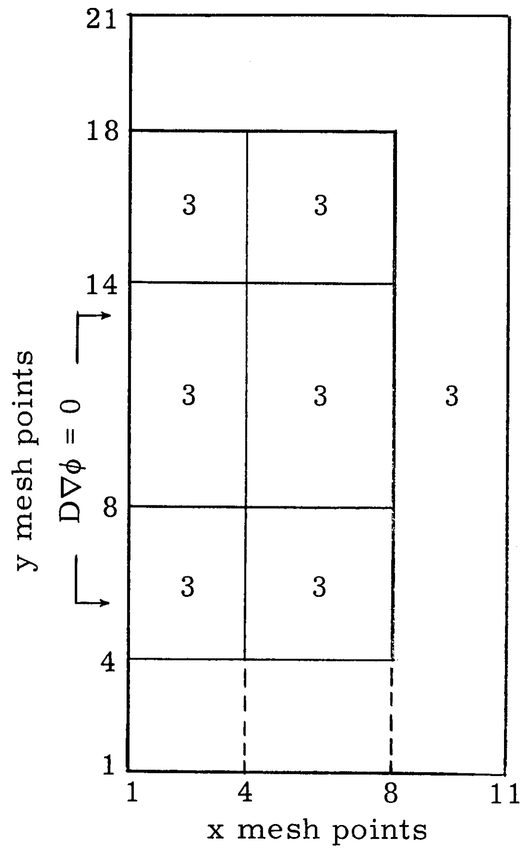

Fig. C.1b. x-y Plane for $0 \leq z \leq 24,\ 136 \leq z \leq 160$ cm

The numbers in the various regions indicate the material number in that region. Only the right half of this reactor is shown since the left half is symmetrical to it.

Precursor Constants:

$\lambda_1 = .08, \quad \beta_1 = .0075, \quad \chi_{11} = 1.0, \quad \chi_{21} = 0.0$

	Group 1	Group 2
v	1.0×10^7	2.0×10^5
χ	1.0	0.0

Material Properties:

Material 1

	Group 1	Group 2
Σ_{tr}	.238095	.833333
Σ_a	.01	.15
ν	2.40	2.40
Σ_f	.0035	.10
$\Sigma_{J \to J+1}$	.01	0.0

Material 2

(Same as Material 1)

Material 3

	Group 1	Group 2
Σ_{tr}	.25461	.666667
Σ_a	.008	.05
ν	2.40	2.40
Σ_f	.0015	.03
$\Sigma_{J \to J+1}$	.01	0.0

Initial Condition:

Critical k_{eff}: 1.06432742

Configuration 3

Number of neutron groups = 4

Number of precursor groups = 1

Geometry:

$\Delta x = \Delta y = \Delta z = 8.0$ cm

height = 120 cm (z-direction)

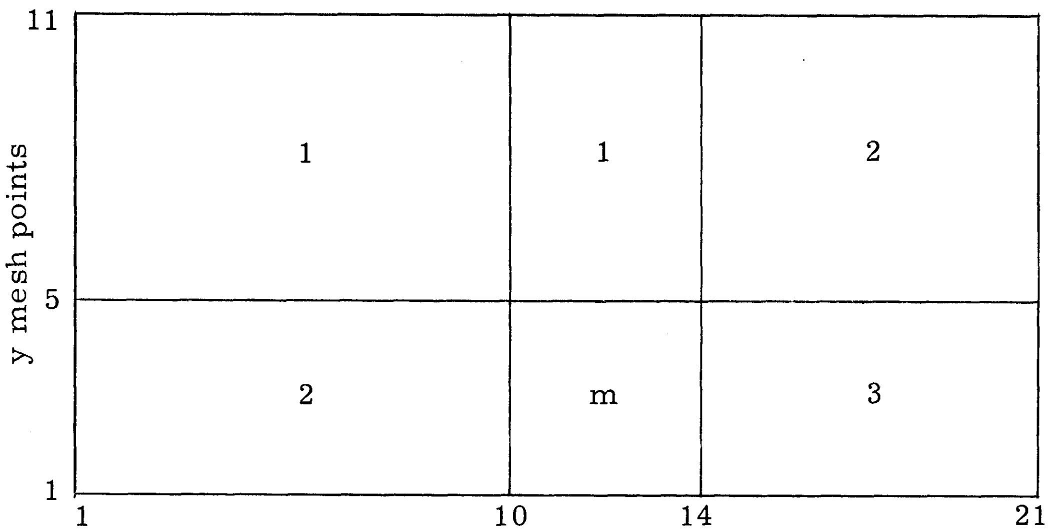

Fig. C.2. x-y Plane for $0 \leq z \leq 120$ cm

$m = 3$ for $0 \leq z \leq 56$ cm

$m = 4$ for $56 \leq z \leq 120$ cm

Precursor Constants:

$\lambda_1 = .08$, $\beta_1 = .0064$, $\chi_{11} = 0.0$, $\chi_{21} = 1.0$, $\chi_{31} = 0.0$,

$\chi_{41} = 0.0$

	Group 1	Group 2	Group 3	Group 4
v	1.0×10^9	1.0×10^8	5.0×10^6	2.0×10^5
χ	0.755	0.245	0.0	0.0

Material Properties:

Material 1

	Group 1	Group 2	Group 3	Group 4
Σ_{tr}	.120	.310	.520	2.050
Σ_a	.00266	.00297	.0359	.655
ν	1.60	1.60	1.60	1.60
Σ_f	.00136	.00197	.0262	.540
$\Sigma_{J \to J+1}$	.0586	.0828	.0850	0.0

Material 2

	Group 1	Group 2	Group 3	Group 4
Σ_{tr}	.100	.240	.400	1.600
Σ_a	.00135	.00140	.0176	.332
ν	1.60	1.60	1.60	1.60
Σ_f	.0007	.0009	.0131	.274
$\Sigma_{J \to J+1}$	.0586	.0828	.0850	0.0

Material 3

	Group 1	Group 2	Group 3	Group 4
Σ_{tr}	.080	.160	.310	1.270
Σ_a	.00077	.00072	.00051	.012
ν	0.0	0.0	0.0	0.0
Σ_f	0.0	0.0	0.0	0.0
$\Sigma_{J \to J+1}$	.0570	.0822	.0847	0.0

Material 4

(Same as Material 3)

Initial Condition:

Critical k_{eff}: 1.06601870

Configuration 4

Number of neutron groups = 2

Number of precursor groups = 1

Geometry:

Height = 300 cm (z-direction)

$\Delta x = 10.0$ cm, $0 \leq x \leq 20$ cm, $50 \leq x \leq 170$ cm, $200 \leq x \leq 220$ cm

$\Delta x = 7.5$ cm, $20 \leq x \leq 50$ cm, $170 \leq x \leq 200$ cm

$\Delta y = 7.5$ cm, $0 \leq y \leq 30$ cm

$\Delta y = 10.0$ cm, $30 \leq y \leq 110$ cm

$\Delta z = 15.0$ cm, $0 \leq z \leq 30$ cm, $270 \leq z \leq 300$ cm

$\Delta z = 16.0$ cm, $30 \leq z \leq 270$ cm

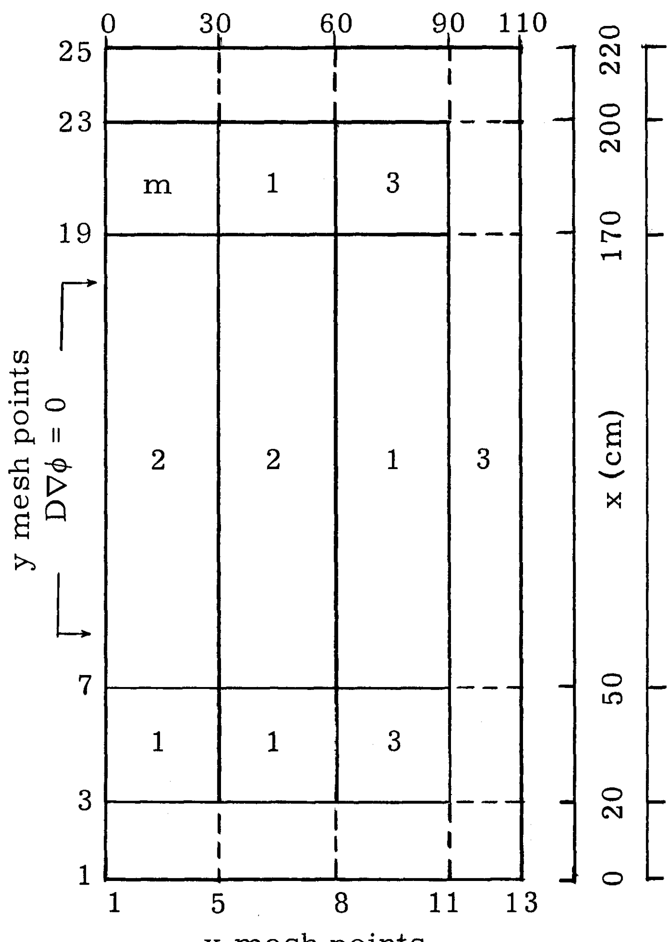

Fig. C.3a. x-y Plane for $30 \leq z \leq 270$ cm

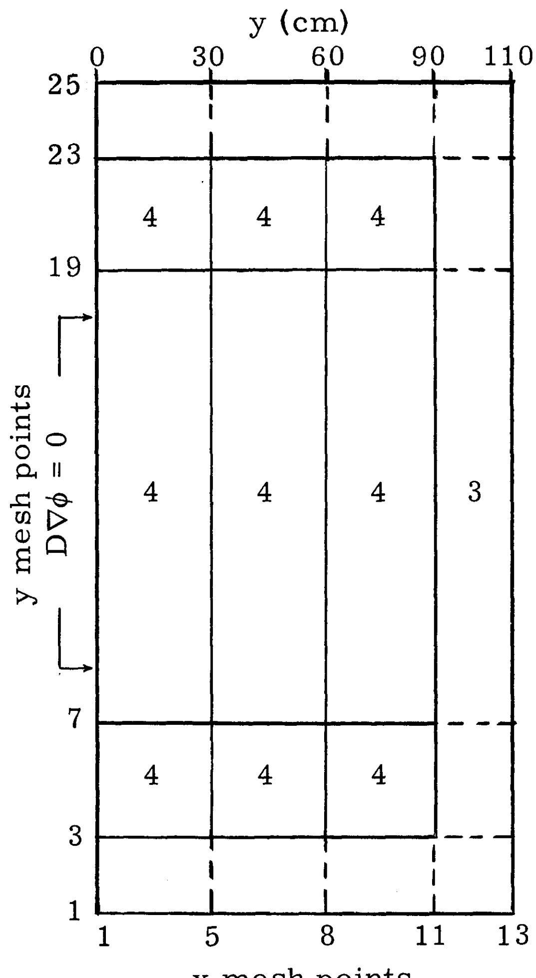

Fig. C.3b. x-y Plane for $0 \leq z \leq 30$ cm, $270 \leq z \leq 300$ cm

$$m = \begin{cases} 5, & \text{for } 30 \leq z \leq 110 \text{ cm} \\ 6, & \text{for } 110 \leq z \leq 190 \text{ cm} \\ 7, & \text{for } 190 \leq z \leq 270 \text{ cm} \end{cases}$$

Precursor Constants:

$\lambda_1 = .08, \quad \beta_1 = .0064, \quad \chi_{11} = 1.0, \quad \chi_{21} = 0.0$

	Group 1	Group 2
v	1.0×10^8	4.4×10^5
χ	1.0	0.0

Material Properties:

Material 1

	Group 1	Group 2
Σ_{tr}	.2246	.8375
Σ_a	.009434	.07345
ν	2.571	2.441
Σ_f	.002437	.05112
$\Sigma_{J \to J+1}$	.01872	0.0

Material 2

	Group 1	Group 2
Σ_{tr}	.2264	.8445
Σ_a	.009223	.06737
ν	2.584	2.442
Σ_f	.002236	.04557
$\Sigma_{J \to J+1}$	.01893	0.0

Material 3

	Group 1	Group 2
Σ_{tr}	.1971	.8685
Σ_a	.0001984	.007207
ν	0.0	0.0
Σ_f	0.0	0.0
$\Sigma_{J \rightarrow J+1}$	.03010	0.0

Material 4

	Group 1	Group 2
Σ_{tr}	.1487	.5490
Σ_a	.0003288	.004677
ν	0.0	0.0
Σ_f	0.0	0.0
$\Sigma_{J \rightarrow J+1}$	.01798	0.0

Material 5

(Same as Material 1)

Material 6

(Same as Material 1)

Material 7

(Same as Material 1)

Initial Condition:

Critical k_{eff}: 1.28041608

Appendix D
THE COMPUTER PROGRAM 3DKIN

The computer program used to conduct the three-dimensional numerical experiments for this thesis has been named 3DKIN. It is written entirely in Fortran IV for IBM System 360 computers. It can be easily converted to run on any computer with a Fortran IV compiler, however.

The program 3DKIN is described in the several sections of this appendix. It is intended that the description given here be adequate for this appendix to serve as a user's manual for the code. A more detailed description would be necessary for anyone wishing to make modifications to the code, however.

Section D.1 discusses the methods used to obtain an initial steady state solution for a problem. Section D.2 then describes the organization of that part of the code used in a subsequent time-dependent calculation. The overlay structure used to reduce core storage requirements and the input/output devices necessary to run 3DKIN are presented in section D.3. A detailed description of the input information for 3DKIN follows in section D.4. Section D.5 lists the card images of the input data for 3DKIN for a sample problem.

D.1 Description of the Steady State Section

Several comments of a general nature concerning 3DKIN should be made before proceeding to the algorithms used to obtain the initial critical flux distribution and k_{eff}. The first concerns the overall organization of the code. It has been written in a modular fashion, where each subroutine or set of subroutines performs one task or several closely-related tasks. This facilitates the division of the code into segments for subsequent use of the overlay feature of OS/360. It also allows additional code options such as new geometries to be added to the code without severely altering existing subroutines.

The second comment concerns the use of directly-addressable core storage for the storage of program variables. The variable dimensioning feature of Fortran IV is used throughout the code. In the MAIN routine, a vector named A is placed in the labeled common area ARAY and given a length which corresponds to the total core area which the user desires to allot to program variable storage. Based on input parameters which describe the size of the problem to be considered, a subroutine called MEMORY computes a series of pointer variables. Each pointer variable indicates a location in A where the first member of a program array will be located. The remaining members of that array are then stored in successive locations in A.

The obvious advantage of this technique is that each program array is dimensioned to exactly the size necessary for each particular problem. Core storage is thus used very efficiently. In addition, the total amount of core storage allotted to program variable storage can be changed merely by recompiling the short MAIN routine after changing two statements.

For both the steady state and time-dependent parts of the code, the total core storage necessary to store program variables for each problem is computed. If this amount exceeds the amount allocated to the vector A in MAIN, another attempt is automatically made to allot storage for program variables. This time, however, the flux and fission source vectors are stored on input/output devices for the steady state section, as are the fluxes and precursor concentrations for the time-dependent section. This greatly reduces the amount of core storage required and allows very large problems to be run.

The remainder of this section describes the program flow in the steady state section. The program entry point is in the MAIN routine. The MAIN routine zeroes out the entire A array and reads in the title card and second card for a particular problem. The second card contains the parameters which completely define the amount of storage required for that problem. Subroutine MEMORY is called to allocate storage for program variables and to determine whether or not input/output devices are required to store several large arrays. If these input/output devices are required, MAIN opens the datasets on these devices. Program control is then passed to subroutine CALLER, which calls the subroutines which control the various first overlay level segments.

Subroutine INPUT is called first to read in the remaining input data for the problem. Subroutine IOEDIT prints out an edited version of the problem description. The flux vector needed to start the iterative solution process is read in either from cards or from a dataset on an input/output device or is generated as a cosine in each dimension. Subroutine FLUXIN performs whichever of these options is requested.

The iterative solution process is controlled by subroutine SSTATE. To detail the form of this iterative process, several equations need to be restated. The time-dependent equation for the group g flux at all points has been given in Eq. (1.4) as

$$\frac{d\vec{\psi}_g}{dt} = \underline{D}_g \vec{\psi}_g + \sum_{g'=1}^{G} \underline{T}_{gg'} \vec{\psi}_{g'} + \sum_{i=1}^{I} \underline{F}_{gi} \vec{C}_i \; . \tag{1.4}$$

To obtain the initial condition, the time derivative is set to zero. Further, given the definitions of χ_g and χ_{gi} in section 1.2, a weighted prompt fission spectrum, χ'_g, can be defined as

$$\chi'_g = (1-\beta)\,\chi_g + \sum_{i=1}^{I} \beta_i \chi_{gi} \; . \tag{D.1}$$

If χ'_g replaces χ_g in Eq. (1.4), the precursor concentration term in it can also be ignored for the steady state calculation.

Several additional matrices need to be defined. Let

$$\underline{T}_{gg'} = \underline{v}_g \underline{V}^{-1} (\chi'_g \underline{F}_{g'} + \underline{R}_{gg'}), \quad g' \neq g \tag{D.2a}$$

$$\underline{T}_{gg} = \underline{v}_g \underline{V}^{-1} (\chi'_g \underline{F}_g - \underline{\Sigma}_g) \tag{D.2b}$$

$$\underline{D}_g = \underline{v}_g \underline{V}^{-1} \underline{D}'_g \; . \tag{D.2c}$$

Here, $\underline{v}_g = v_g \underline{I}$ and $\underline{V}$ is the diagonal matrix of volumes associated with each mesh point. $\underline{F}_{g'}$ is a diagonal matrix containing the $v_{g'} \Sigma_{fg'}$ term for each mesh volume. The matrix $\underline{R}_{gg'}$ is also diagonal and describes the scattering from group g' to g in each mesh volume. Finally, $\underline{\Sigma}_g$ contains the absorption and out-scattering terms for group g at each mesh volume.

The form of Eq. (1.4) to be solved for the initial condition becomes

$$\underline{v}_g \underline{V}^{-1}(\underline{D}'_g - \underline{\Sigma}_g)\vec{\psi}_g + \underline{v}_g \underline{V}^{-1}\left(\sum_{g' \neq g}^{G} \underline{R}_{gg'}\vec{\psi}_{g'} + \frac{\chi'_g}{k_{eff}} \sum_{g'=1}^{G} \underline{F}_{g'}\vec{\psi}_{g'} \right) = \vec{0},$$

$$(1 \leq g \leq G). \qquad \text{(D.3)}$$

In 3DKIN, only downscattering is allowed so that $\underline{R}_{gg'} = \underline{0}$ for $g' > g$. Equation (D.3) can be reduced to

$$(-\underline{D}'_g + \underline{\Sigma}_g)\vec{\psi}_g = \sum_{g'=1}^{g-1} \underline{R}_{gg'}\vec{\psi}_{g'} + \frac{\chi'_g}{k_{eff}} \sum_{g'=1}^{G} \underline{F}_{g'}\vec{\psi}_{g'} . \qquad \text{(D.4)}$$

In 3DKIN, Eq. (D.4) is solved by a two-level iterative process. This is the standard inner iteration-outer iteration method.[27] Let the inner iteration index be m and the outer iteration index be ℓ. The inner iterations involve solving the equation

$$(-\underline{D}'_g + \underline{\Sigma}_g)\vec{\psi}_g^{\ell+1} = \sum_{g'=1}^{g-1} \underline{R}_{gg'}\vec{\psi}_{g'}^{\ell+1} + \frac{\chi'_g}{\sigma^\ell}\vec{S}^\ell \qquad \text{(D.5)}$$

for each group, starting with g=1. Here, $\vec{S}^\ell$, the fission source vector, and σ^ℓ have been obtained from

$$\sigma^\ell = \frac{\left\| \sum_{g=1}^{G} \underline{F}_g \vec{\psi}_g^\ell \right\|_1}{\left\| \sum_{g=1}^{G} \underline{F}_g \vec{\psi}_g^{\ell-1} \right\|_1}, \qquad \text{(D.6)}$$

$$\vec{S}^\ell = \frac{\mu}{\sigma^\ell} \sum_{g=1}^{G} \underline{F}_g \vec{\psi}_g^\ell + \frac{(1-\mu)}{\sigma^\ell} \sum_{g=1}^{G} \underline{F}_g \vec{\psi}_g^\ell . \qquad \text{(D.7)}$$

Here, μ is an input fission source overrelaxation parameter bounded by $1 \leq \mu \leq 2$. The outer iteration consists of the computation of σ^ℓ and an $\vec{S}^\ell$, used to start a new set of inner iterations.

The inner iterations in 3DKIN are carried out by a one-line successive overrelaxation method. Lines of fluxes in the x-direction are overrelaxed successively across each z-plane of mesh points, starting with the bottom z-plane. An optimum overrelaxation parameter is computed for each group, using a method prescribed in Ref. 27. The iterative process continues on a particular group until convergence is obtained for that group, where convergence is defined as

$$\max_{l,j,k} \left| \frac{\phi^{m}_{g,l,j,k} - \phi^{m-1}_{g,l,j,k}}{\phi^{m}_{g,l,j,k}} \right| \leq \epsilon_2 . \tag{D.8}$$

The parameter ϵ is input by the user, as is a parameter m_{max}. If the condition (D.8) is not satisfied for $m \leq m_{max}$, the iterative process is stopped for that group automatically for that outer iteration.

As can be seen from Eq. (D.6), the L_1 norm is used as an indication of the total solution change during an outer iteration. In an attempt to speed convergence of the outer iterations, the fission source vector, $\vec{S}^\ell$, is overrelaxed in a rather crude fashion. The entire iterative process is completed after the ℓ^{th} outer iteration if condition (D.8) has been satisfied for all groups during that outer iteration and if

$$\left| 1.0 - \sigma^\ell \right| \leq \epsilon_1 . \tag{D.9}$$

At this point, k_{eff} is computed from

$$k_{eff} = \prod_{n=1}^{\ell} \sigma^n . \tag{D.10}$$

Before starting the iterative process just described, subroutine SSTATE calls subroutine SETUP1 to compute the necessary coefficients. SETUP1 uses subroutine COEF1 to do this.

SSTATE also calls subroutine ORPEST to compute the groupwise optimum overrelaxation parameters. SSTATE computes the fission and scattering source for each group during an outer iteration. Subroutine INNER0 or INNER1 is called to carry out the actual inner iterations for the groups. INNER0 is used if all program variables are stored in core, while INNER1 is used if the flux and fission source vectors are stored on input/output devices. SSTATE completes the outer iteration by computing a new estimate of σ and overrelaxing the fission source vector. It also tests for convergence of the outer iterations. Subroutine SSTOUT prints out a one-line summary of each outer iteration and saves the converged fluxes if requested.

Two additional features of the steady state section of 3DKIN are worthy of note, although they are invisible to the user of 3DKIN. The first is an additional technique used to accelerate convergence of the inner iterations. Before the inner iterations are started for group g during outer iteration $\ell+1$, the quantities

$$\alpha_1 = \left\| \sum_{g'=1}^{g-1} \underline{R}_{gg'} \vec{\psi}_{g'}^{\ell+1} + \frac{\chi'_g}{\sigma^\ell} \vec{S}^\ell \right\|_1$$

and

$$\alpha_2 = \left\| (-\underline{D}'_g + \underline{\Sigma}_g) \vec{\psi}_g^\ell \right\|_1$$

are computed. The vector $\vec{\psi}_g^\ell$ is multiplied by the ratio (α_1/α_2), and the result is used as an initial guess for the inner iterations for group g. This has the effect of scaling the initial guess so that the neutron balance is satisfied in an integral sense when the inner iterations are started. This so-called group rebalancing is carried out by subroutines GRBAL0 and GRBAL1, which are called by INNER0 and INNER1, respectively.

The second feature is the manner in which the coefficients for Eqs. (D.4) are stored in 3DKIN for the x-y-z geometry option. The manner in which planes are passed through the parallelepiped of interest to create the three-dimensional fine mesh has been presented in Appendix A. The only restriction placed on these planes at that time was that every boundary of a homogeneous material region must lie on a fine-mesh plane.

In 3DKIN, an additional restriction is introduced. Each of the fine-mesh planes which has a homogeneous material region boundary coincident on any part of it becomes a coarse-mesh plane. Between two successive coarse-mesh planes in a particular direction, all fine-mesh planes parallel to these coarse-mesh planes must be equidistant.

The reactor of interest is thus divided into a three-dimensional array of rectangular parallelepipeds by the coarse-mesh planes. These rectangular parallelepipeds are hereafter referred to as material regions. Within a given material region, only one material is present. Additionally, fine-mesh spacings are constant across that material region for each of the three directions.

Each material region has a total of 26 faces, edges, and corners associated with it. Thus, regardless of how many fine-mesh points lie within or on its boundaries, only 27 sets of coefficients need to be computed and stored. The extra set is for all of the fine-mesh points which lie within the boundaries of the material region.

Because of the manner in which faces, edges, and corners are shared by more than one material region, however, an average of only 8 sets need to be associated with each material region. This assumes that the right, upper, and back outer boundaries of the parallelepiped as shown in Fig. A.1 have homogeneous Dirichlet boundary conditions.

In 3DKIN, a so-called problem region number is assigned to each set of coefficients. A three-dimensional array, called a problem region map, is created, with one entry per fine-mesh point. In this problem region map, all fine-mesh points which have the same set of coefficients are assigned the same unique problem region number. Coefficients are computed and stored by problem region number, and the problem region map is used to obtain the proper set of coefficients to be used at a particular fine-mesh point.

The advantages of this method are two-fold. No coefficients ever have to be recomputed during the entire steady state calculation, and each fine-mesh point has a set of coefficients correct for it. At the same time, the amount of storage necessary to contain the coefficients is reduced drastically over that required if a set of coefficients were computed and stored for each fine-mesh point.

D.2 Description of Time-Dependent Section

The program flow for the time-dependent section of 3DKIN is much less complicated than that for the steady state section. This is primarily due to the simplicity of the NSADE algorithm. When the initial condition has been computed, SSTATE returns program control to CALLER. CALLER calls subroutine FLUXTR, which writes the converged fluxes out on an input/output device. CALLER then calls subroutine TIMDEP, which controls the remainder of the time-dependent section.

Subroutine TIMDEP first redefines several coefficients in each problem region. It then calls subroutine DELAYS, which reads the fluxes back in from the input/output device and computes the corresponding pointwise initial precursor concentrations. After zeroing out the frequency array and dividing the various $\nu\Sigma_f$ values by the critical value of k_{eff}, the main time-dependent loop in TIMDEP is entered.

Within this main loop, time is divided into a series of time zones. Within each time zone, a number of materials are allowed to have properties which undergo a step change at the beginning of the time zone and/or a ramp change throughout the time zone. Subroutine TIMINP reads in the data describing each of these time zones.

Within each time zone, subroutine CHANGE is called whenever necessary to recompute coefficients which vary with time. The coefficients are recomputed consistent with the problem region concept. Coefficients are recomputed only for those problem regions which have time-varying properties.

For the case where all problem variables are stored in core, the initial $e^{\Omega h}$ transformation and forward sweep of the spatial mesh for all groups is performed in subroutine STEPA0 for each time step. Subroutine STEPB0 performs the reverse sweep and the second $e^{\Omega h}$ transformation for each time step. Subroutine FREQ0 computes the frequencies for the next time step according to Eq. (2.8). For the case where the fluxes and precursor concentrations are stored on input/output devices, subroutines STEPA1, STEPB1, and FREQ1 perform the same functions as their similarly-named counterparts.

At regular intervals, the fluxes at a number of specified test points are printed out. At the end of each time zone, the entire flux and precursor vector can be printed out if requested. These printouts are obtained from the subroutine TIMOUT.

D.3 Overlay Structure and Input/Output Devices for 3DKIN

Two levels of overlay are used in 3DKIN. There are a total of 11 segments. The overlay structure is shown in Fig. D.1.

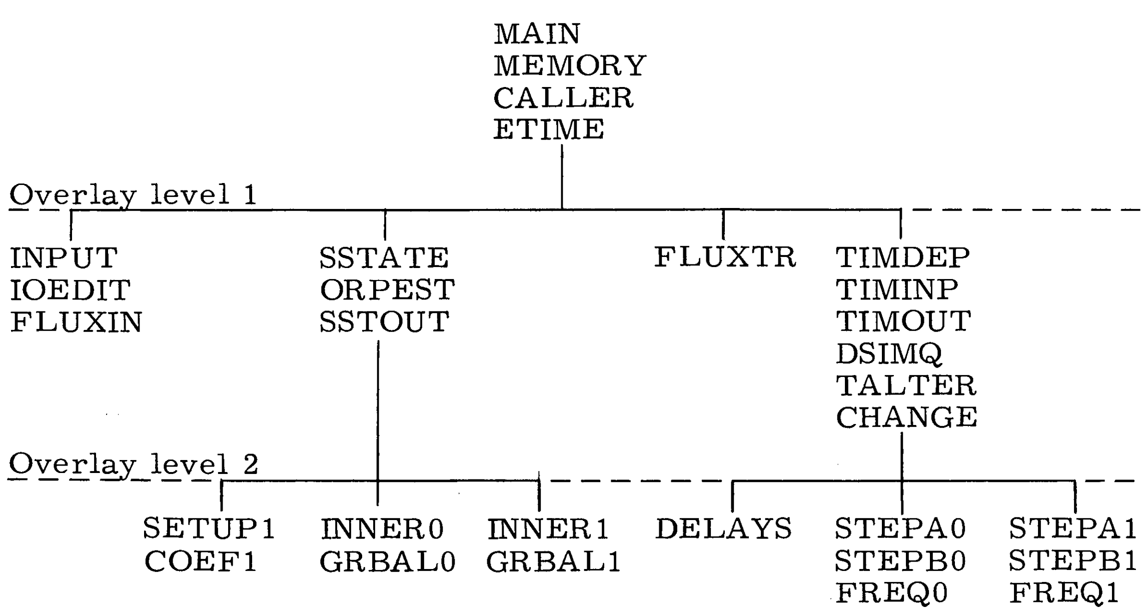

Fig. D.1. Overlay Structure for 3DKIN

Card input to 3DKIN is read in on symbolic device 5, while output to the printer is on device 6. If the option where the fluxes are punched onto cards is requested, the card punch is specified as device 7.

Up to seven sequential datasets on different symbolic devices may be required by 3DKIN. These datasets may each be placed on a separate magnetic tape drive, or they may be placed on one or more disk drives. The disk drives are preferable because their use generally results in faster execution times.

If the option is requested where steady state fluxes are to be stored on an input/output device between runs, this dataset is placed on symbolic device 8. Additionally, symbolic devices 11 and 12 are required for every run in which a time-dependent calculation is to be made. These datasets are used for scratch purposes only.

If the problem is large enough to require that several program vectors be stored on input/output devices, symbolic devices 11 and 12 are required during the steady state calculation as well. The fluxes for each group are spooled back and forth from one to the other during the inner iterations for that group.

In addition, four more symbolic devices are required for scratch purposes for these large problems. During the steady state calculation, old and new fission source vectors alternate on devices 1 and 2. During the time-dependent calculation, the flux for the group used in the frequency calculation is saved from one time step to the next, alternately, on these two devices. Devices 3 and 4 alternate in storing the complete flux vector (and precursors as well during the time-dependent calculation) for both sections of the code. All of these seven datasets are written with unformatted write statements. Table D.1 summarizes the usage of these datasets.

Table D.1. Input/Output Symbolic Devices

Device Number	Logical Record Length		Number of Records		When Used
	Steady State	Time-Dependent	Steady State	Time-Dependent	
1	L * J	L * J	K	K	IOPT = 1
2	L * J	L * J	K	K	IOPT = 1
3	L * J	L * J * (G+I)	K * G	K	IOPT = 1
4	L * J	L * J * (G+I)	K * G	K	IOPT = 1
8	L * J	L * J	K * G	K * G	When fluxes saved
11	L * J	L * J	K	K * G	Always
12	L * J	L * J	K	K * G	Always

The variables L, J, and K are the number of fine-mesh x-planes, y-planes, and z-planes, respectively. In order to minimize execution time, symbolic devices 1, 3, and 11 should be placed on different disk drives from symbolic devices 2, 4, and 12, respectively.

D.4 Description of Input for 3DKIN

The only geometry currently available in 3DKIN is x-y-z rectangular geometry, as shown in Fig. A.1. For both steady state and time-dependent sections, the right, back, and top faces of the rectangular parallelepiped must have homogeneous Dirichlet boundary conditions.

In the steady state section, the left, front, and bottom faces may each have homogeneous Dirichlet or Neumann boundary conditions specified, independent of what condition is specified for the other two faces. In the time-dependent section, however, the bottom face must always have the homogeneous Dirichlet condition. If only one face is to have a homogeneous Neumann condition, it must be the left face. If quarter-core symmetry is desired, both left and front faces are specified to have a homogeneous Neumann boundary condition.

At the time that the input description of a problem is formulated, an estimate of the amount of core required for program variable storage can be made. Equation (D.11) gives the total number of double precision (64-bit) words required on an IBM System 360 computer in the vector A for each problem. The variables in the equation are defined in the input description.

$$\begin{aligned}\text{Min. length of A} = {} & \text{NNG} * [3+(4+\text{NDNSCT})*\text{NMAT}+\text{NDG}] + 2 * \text{NDG} \\ & + \text{IM} + \text{JM} + \text{KM} + 3 * (\text{IRM}+\text{JRM}+\text{KRM}+1)/2 \\ & + 8 * \text{IRM} * \text{JRM} * \text{KRM} * [6*\text{NNG}+(\text{NNG}-1) \\ & * \text{NDNSCT}+1] + (\text{IRM}*\text{JRM}*\text{KRM}+3)/4 \\ & + (\text{IM}*\text{JM}*\text{KM}+3)/4 + V_{core} \, . \end{aligned} \tag{D.11}$$

For the steady state section,

$$\begin{aligned} V_{core} = {} & 3 * \text{IM} + 5 * \text{NNG} + \text{IM} * \text{JM} * \text{KM} + \\ & (1-\text{IOPT}) * [\text{IM}*\text{JM}*\text{KM}*(\text{NNG}+2)+5] + \\ & \text{IOPT} * (5*\text{IM}*\text{JM}+5) \, . \end{aligned}$$

For the time-dependent section,

$$V_{core} = IM * JM * KM + (1-IOPT) * [IM*JM*KM*(NNG+NDG+1)+5] + IOPT * [IM*JM*(3*NNG+3*NDG+2)+2].$$

Setting IOPT = 0 gives the minimum length of A required if all variables are to be stored in core. Likewise, setting IOPT = 1 gives the core storage requirement for the option where several vectors are stored on input/output devices.

Using the Fortran H compiler with optimization level 2 and the level 18.6 version of OS-MVT for the IBM 360/65, a total of 77,500 bytes are required to store 3DKIN in core, exclusive of the number of bytes allocated to the vector A. In addition, when the code is actually executed, some additional core is needed for input/output device buffers. With 46,000 8-byte words allocated to A and with about 12,000 bytes allocated to buffers, 3DKIN requires 458,000 bytes of core. A load module of this size was necessary to run Test Case 4 in Chapter 3.

What follows is a card-by-card description of the input for 3DKIN.

Card Type 1 FORMAT(20A4)

Columns 1-80: (ITITLE(I), I=1, 20). This is the alphanumeric problem title.

Card Type 2 FORMAT(20I4)

Columns 1-4: NNG. This is the number of prompt neutron groups.

Columns 5-8: NDG. This is the number of precursor groups.

Columns 9-12: NTG. This is the number of the group to be used in the frequency calculation for the time-dependent section.

Columns 13-16: NDNSCT. This is the maximum number of down-scatter groups for any of the neutron groups. No upscattering is allowed in 3DKIN.

Columns 17-20: NMAT. The code expects to read in a total of NMAT macroscopic cross-section sets. These sets are numbered consecutively, from 1 to NMAT.

Columns 21-24, 25-28, 29-32: IM, JM, KM. These variables give, respectively, the number of fine-mesh x-planes, y-planes, and z-planes. The outer boundary planes are included.

Columns 33-36, 37-40, 41-44: IRM, JRM, KRM. These variables indicate the number of coarse-mesh zones in the x-, y-, and z-direction, respectively.

Columns 45-48, 49-52, 53-56: NXTP, NYTP, NZTP. These are the number of x, y, and z points, respectively, that are to be used in printing out fluxes during the time-dependent calculation. Every IPRSTP steps, a total of NXTP*NYTP*NZTP points will have their flux values printed out.

Columns 57-60: NSTEAD. If NSTEAD = 0, only a time-dependent calculation will be performed. The input fluxes will be taken as the initial condition. If NSTEAD = 1, a steady state calculation will first be performed. If the solution converges within NOIT outer iterations, a time-dependent calculation will follow. If NSTEAD = 2, only a steady state calculation will be performed.

Columns 61-64: IFLIN. If IFLIN = 0, the initial flux will be generated by 3DKIN as a cosine in each direction for each group. If IFLIN = 1, the initial fluxes are to be input on cards. If IFLIN = 2, the initial fluxes are to be read in as a sequential dataset from device 8.

Columns 65-68: IFLOUT. This variable applies only to the output of fluxes at the end of a steady state calculation. If IFLOUT = 0, no fluxes will be output. If IFLOUT = 1, the fluxes will be printed out. If IFLOUT = 2, the fluxes will be printed and also punched onto cards in a 5D16.10 format. If IFLOUT = 3, the fluxes will be printed and also written on device 8 as a sequential dataset. If IFLOUT = 4, the fluxes are only written on device 8.

Columns 69-72: IGEOM. This is the geometry indicator. At present, IGEOM = 1 gives x-y-z geometry, the only option available.

Columns 73-76: IETIME. If IETIME > 0, the outer iteration completed after accumulated computing time exceeds IETIME will be the last. The fluxes at that point are output as indicated by IFLOUT, and the program stops. If IETIME = 0, it is ignored.

Card Type 3 FORMAT(E16.10, 4X, 3E10.4, 3I4)

Columns 1-16: EFFK. This is the initial estimate of k_{eff}. If it is not in the range $.1 \leq k_{eff} \leq 10.0$, it is set to 1.0.

Columns 21-30: ORFP. This is the parameter used to overrelax the fission source vector, as in Eq. (D.7). It should be in the range $1.0 \leq ORFP \leq 2.0$.

Columns 31-40: EPS1. This is the eigenvalue convergence parameter, ϵ_1, from Eq. (D.9).

Columns 41-50: EPS2. This is the flux convergence parameter, ϵ_2, from Eq. (D.8).

Columns 41-44: NOIT. This is the maximum number of outer iterations allowed in the steady state section. If convergence has not been obtained after NOIT outer iterations, the eigenvalue estimate is printed, the fluxes at that time are output as indicated, and the program is stopped. Provided the fluxes have been saved on cards or on device 8, the latest k_{eff} can be input to a new run with these fluxes and the calculation restarted.

Columns 45-48: NIIT. This is the maximum number of inner iterations per group per outer iteration.

Columns 49-52: NPIT. If the flux and fission source vectors are stored on input/output devices (IOPT=1), then the fluxes for a group are recomputed across each plane a total of NPIT times before going to the next plane during the inner iterations.

Card Type 3′ FORMAT(8E10.4)

Use as many cards as are necessary.

Columns 1-10, 11-20, . . . : (OMEG(NG),NG=1,NNG). These are estimates of the overrelaxation parameters for the inner iterations. If any OMEG(NG) is in the range $.95 \leq \text{OMEG(NG)} \leq 1.05$, all of them will be computed by 3DKIN to be the optimum values. Once the optimum values are known, they can be input and the calculation thus avoided.

<u>Card Type 4</u> FORMAT(I5, 5(I5,E10.4)/5(I5, E10.4))

One set of these cards is needed for each of the three directions.

First set –

<u>Columns 1-5: NLBC</u>. This is the boundary condition at x = 0. NLBC = 0 indicates a zero flux (homogeneous Dirichlet) condition, while NLBC = 1 indicates a zero current (homogeneous Neumann) condition.

<u>Columns 6-10, 11-20; 21-25, 26-35; . . . : (IBP(IR),HX(IR),IR=1,IRM)</u>. IBP(IR) is the right x fine-mesh plane number for the IRth x coarse-mesh region. HX(IR) is the total x-width for that region in centimeters. Additional cards may be used for these pairs of boundary planes and widths. If the last card has five pairs on it, a blank card must follow it.

Second set –

<u>Columns 1-5: NFBC</u>. This is the boundary condition at y=0. For the steady state section, it can be either 0 (zero flux) or 1 (zero current). It can be 1 only if NLBC = 1 for the time-dependent section.

<u>Columns 6-10, 11-20; 21-25, 26-35; . . . : (JBP(JR), HY(JR), JR=1, JRM)</u>. These are the pairs of back fine-mesh y-planes and total y-widths for the y coarse-mesh zones.

Third set –

<u>Columns 1-5: NBBC</u>. This is the boundary condition at z=0. Either a 0 or a 1 can be used for a steady state calculation, but only a zero flux boundary condition is allowed here for the time-dependent calculation.

<u>Columns 6-10, 11-20; 21-25, 26-35; . . . : (KBP(KR), HZ(KR), KR=1, KRM)</u>. These are the pairs of upper fine-mesh z-planes and total z-widths for the z coarse-mesh zones.

Card Type 5 FORMAT(20I4)

Use as many cards as necessary.

Columns 1-4, 5-8, . . . : (IXTP(I), I=1, NXTP), (IYTP(I), I=1, NYTP), (IZTP(I), I=1, IZTP). These are the points at which fluxes will be printed out every IPRSTP steps during the time-dependent calculation.

Card Type 6 FORMAT(20I4)

One set of cards is required for each coarse-mesh z-region. Use as many cards as necessary for each set, with 20 values on each card.

Columns 1-4, 5-8, . . . : ((MMAP(IR, JR, KR), IR=1, IRM), JR=1, JRM). These are the material numbers assigned to each material region in the KRth coarse-mesh z-region.

Card Type 7 FORMAT(6E12.6)

Columns 1-12, 13-24, . . . : (V(NG), NG=1, NNG). These are the group velocities in cm/sec.

Card Type 8 FORMAT(6E12.6)

Columns 1-12, 13-24, . . . : (XI(NG), NG=1, NNG). This is the prompt fission spectrum.

A set of NNG card type 9's and as many card type 10's as are necessary is input as a package for each material NM, $1 \leq NM \leq NMAT$. The sets start with material 1 and proceed consecutively to material NMAT.

Card Type 9 FORMAT(4E12.6)

Columns 1-12: XNU(NM, NG). This is ν for group NG.

Columns 13-24: SIGF(NM, NG). This is Σ_f for group NG in cm^{-1}.

Columns 25-36: SIGR(NM, NG). This is Σ_a for group NG in cm^{-1}.

Columns 37-48: SIGT(NM, NG). This is Σ_{tr} for group NG in cm^{-1}.

In each set, the NNG card type 9's are arranged consecutively from group 1 to group NNG.

Card Type 10 FORMAT(6E12.6)

Columns 1-12, 13-24, ... : ((SIGS(NM, NG, NDN), NDN=1, NDNSCT), NG=1, NGX). This is $\Sigma_{s_{g'g}}$ for g' = NG+1 to g' = NG+NDNSCT for each group g = NG. NGX = NNG-1, so no values are read in for group NNG.

Card Type 11 FORMAT(6E12.6)

One or more card type 11 is required for each precursor group. Begin on a new card for each precursor group.

Columns 1-12: ALAM(ND). This is the λ for precursor group ND in sec^{-1}.

Columns 13-24: BETA(ND). This is β for each precursor group ND.

Columns 25-36, 37-48, ... : (XIP(NG, ND), NG=1, NNG). This is χ_{gi} for all groups g, $1 \leq g \leq NNG$, for precursor group ND.

Card Type 12 FORMAT(5E16.10)

These cards are needed only if IFLIN = 1. There are a total of NNG * KM sets of card type 12's required then. Each set begins on a new card and contains the fluxes for one z-plane and one group. The sets are arranged from plane 1 to plane KM for each group, with those for group 1 coming first.

Columns 1-16, 17-32, . . . : ((PSI(NG, I, J, K), I=1, IM), J=1, JM). These are the fluxes at all points on z-plane KR for group NG.

For a time-dependent calculation, a set of one card type 13, NNG * ISTPCH card type 14's, and NNG * ILINCH card type 15's are needed for each time zone.

Card Type 13 FORMAT (6I5, 3E12.5)

Columns 1-5: LASZON. If > 0, this is the time zone number. If LASZON = 0, this is the last time zone for this problem.

Columns 6-10: ISTPCH. If ISTPCH = 0, no step change in any material properties will occur at the beginning of this time zone. If ISTPCH > 0, then a total of ISTPCH materials have one or more properties which undergo step changes at the beginning of this time zone.

Columns 11-15: ILINCH. ILINCH indicates the total number of materials in which one or more properties will vary as a linear function of time over this time zone.

Columns 16-20: IPRSTP. During the time-dependent calculation, the fluxes at NXTP * NYTP * NZTP points are printed out every IPRSTPth step.

Columns 21-25: ICHHT. This variable is not used at present.

Columns 26-30: IFLOUT. If IFLOUT = 0, fluxes at only the test points are printed out at the end of this time zone. If IFLOUT = 1, the entire flux and precursor vector is printed out at the end of this time zone.

Columns 31-42: HMIN. The value of h (= $\Delta t/2$) to be used throughout this time zone is given here in sec.

Columns 43-54: HMAX. This variable is not used at present.

Columns 55-66: TEND. This is the time at the end of this time zone in sec. It should be an integer multiple of Δt.

Card Type 14 FORMAT (I5, 5X, 5E12.5)

For each material which has a property undergoing a step change, the NNG card type 14's are ordered by group, from group 1 to group NNG. There is a total of NGG*ISTPCH card type 14's in a time zone set.

Columns 1-5: MNSCH(I). This is the material number for which this change takes place.

Columns 11-22: DELSFS(MN, NG). This is the step change in SIGF(MN, NG) for this time zone.

Columns 23-34: DELSRS(MN, NG). This is the step change in SIGR(MN, NG) for this time zone.

Columns 35-46: DELSTS(MN, NG). This is the step change in SIGT(MN, NG) for this time zone.

Columns 47-58: DELS1S(MN, NG). This is the step change in SIGS(MN, NG, 1) for this time zone.

Columns 59-70: DELS2S(MN, NG). This is the step change in SIGS(MN, NG, 2) for this time zone. It is necessary only if NDNSCT $\geq$ 2.

The MN above corresponds to the value of MNSCH(I) for this card. At the present time, this option is limited to problems having 4 groups or less. Also, the maximum number of materials which can be changed in each time zone is five. However, both of these limitations can be changed by altering several COMMON statements in the code.

Card Type 15 FORMAT (I5, 5X, 5E12.5)

For each material which has a property undergoing a linear variation, the NNG card type 15's are ordered by group, from group 1 to group NNG. There are a total of NNG * ILINCH card type 15's in a time zone set.

Columns 1-5: MNLCH(I). This is the material number for which this change takes place.

Columns 11-22: DELSFL(MN, NG). This is the total amount by which SIGF(MN, NG) is to vary over this time zone.

Columns 23-34: DELSRL(MN, NG). This is the total amount by which SIGR(MN, NG) is to vary over this time zone.

Columns 35-46: DELSTL(MN, NG). This is the total amount by which SIGT(MN, NG) is to vary over this time zone.

Columns 47-58: DELS1L(MN, NG). This is the total amount by which SIGS(MN, NG, 1) is to vary over this time zone.

Columns 59-70: DELS2L(MN, NG). This is the total amount by which SIGS(MN, NG, 2) is to vary over this time zone. It is required only if NONSCT $\geq$ 2.

The MN above corresponds to the value of MNLCH(I) for this card. The limitations concerning number of groups and number of materials apply to card type 15 as they do to card type 14.

Card Type 16 FORMAT (I4)

Columns 1-4: If the number 9999 is placed in these 4 columns, this problem is the last problem in this computer run. If any sequence of numbers other than 9999 is placed here, another problem may be placed immediately after this card. Each problem must have a complete set of input data.

D.5 Input for Sample Problem

On the pages that follow, the data for running a problem on 3DKIN are presented in card image format. This sample problem is actually the data for Test Case 2. For the steady state calculation, the initial flux guess is generated by 3DKIN. A total of 120 outer iterations are allowed. The time-dependent calculation is set to run out to .3 seconds with a Δt of .001 second. This problem requires about two hours of running time on an IBM 360/65.

```
                    FIRST THREE LINES NOT 3DKIN INPUT
                 RIGHT DIGIT OF EACH NUMBER IS OVER COLUMN
1       10        20        30        40        50        60        70        80
   THESIS CASE 3: 3-D VERSION OF TWIGLE PROBLEM WITH HALF CORE SYMMETRY
  2   1   2   1   3  11  21  21   3   5   3   3   5   3   1   0   3   1   0
1.0                 1.4000D 001.0000D-091.0000D-08 200  10   1
1.0       1.0
   1     42.4000D 01     83.2000D 01    112.4000D 01
   0     42.4000D 01     83.2000D 01    144.8000D 01    183.2000D 01    212.4000D 01

   0     42.4000D 01    181.1200D 02    212.4000D 01
  1   6   9   3   6  11  16  19   3  11  19
  3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
  3   3   3   2   1   3   3   2   3   2   1   3   3   3   3
  3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
1.000000D 072.000000D 05
1.0       0.0
2.400000D 003.500000D-031.000000D-022.380952D-01
2.400000D 001.000000D-011.500000D-018.333333D-01
1.000000D-020.0
2.400000D 003.500000D-031.000000D-022.380952D-01
2.400000D 001.000000D-011.500000D-018.333333D-01
1.000000D-020.0
2.400000D 001.500000D-038.000000D-032.564103D-01
2.400000D 003.000000D-025.000000D-026.666667D-01
1.000000D-020.0
8.000000D-027.500000D-031.000000D 000.0
    1    0    1   10    0    15.000000D-045.000000D-042.000000D-01
    1         0.00000D 00 0.00000D 00 0.00000D 00 0.00000D 00 0.00000D 00
    1         0.00000D 00-0.00450D 00 0.00000D 00 0.00000D 00 0.00000D 00
    2    0    0   10    0    02.500000D-042.500000D-042.100000D-01
    0    0    0   10    0    15.000000D-045.000000D-043.000000D-01
9999
```

Appendix E

SOURCE LISTING OF 3DKIN

```
C     **********   STATUS OF 3DKIN AS OF JULY 28, 1971  ************
C
C          ALL FEATURES OF 3DKIN AS DESCRIBED IN APPENDIX D HAVE BEEN
C     TESTED AND ARE WORKING EXCEPT FOR THE FOLLOWING ITEMS:
C          1.  THERE IS A BUG SOMEWHERE IN THE STEADY STATE SECTION FOR
C              THE OPTION WHERE FLUX AND FISSION SOURCE VECTORS ARE
C              STORED ON INPUT/OUTPUT DEVICES (IOPT=1).
C          2.  THE SUBROUTINES FOR THE TIME-DEPENDENT SECTION WHICH
C              PERFORM THE FORWARD AND BACKWARD SWEEP AND CALCULATE
C              NEW FREQUENCIES WHEN IOPT=1 (STEPA1,STEPB1,FREQ1) HAVE
C              NOT BEEN THOROUGHLY TESTED AND ARE NOT INCLUDED IN THIS
C              LISTING.
C          3.  THE QUARTER-CORE SYMMETRY OPTION (NLBC=1,NFBC=1) FOR THE
C              TIME-DEPENDENT SECTION HAS A BUG IN IT.
C
C
C          VARIABLE DIMENSIONING IS USED THROUGHOUT 3DKIN.  SPACE FOR ALL
C     ARRAYS IS ALLOCATED IN THE VECTOR A IN LABELLED COMMON ARAY.  A
C     NUMBER OF POINTERS ARE COMPUTED IN SUBROUTINE MEMORY WHICH INDICATE
C     THE LOCATIONS IN A WHERE EACH OF THE FIRST ELEMENTS OF THE ARRAYS
C     ARE STORED.  THE POINTERS ARE NAMED SO THAT EACH CONSISTS OF THE
C     LETTER L PREFIXED TO THE ARRAY NAME.
C
C
C     MAIN PROGRAM FOR 3DKIN .
        IMPLICIT REAL*8 (A-H,O-Z)
        INTEGER*2 MMAP,NPRMP
       COMMON/INTS/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,
      1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),
      2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,
      3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,
      4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX
       COMMON/POINT/LV,LXI,LXIM,LXNU,LSIGF,LSIGR,LSIGT,LSIGS,LALAM,LBETA,
      1LXIP,LX,LY,LZ,LHX,LHY,LHZ,LIBP,LJBP,LKBP,LDD1,LDD2,LDD3,LDD4,LDD5,
      2LDD6,LDD7,LVO,LMMAP,LNPRMP,LPSI,LP1,LP2,LP3,LFRO,LFRN,LFO,LFN,LSRC
      3,LWA,LGA,LSOLN,LOMEG,LXFISS,LXINSC,LXREM,LXLEK,LTOT,LPSO,LW,LPO,LW
```

```
     41
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT
      COMMON/ARAY/A(46000)
      CALL ETIME
      IASIZE=46000
   99 DO 100 I=1,IASIZE
      A(I)=0.0D0
  100 CONTINUE
      IOPSI=8
      IODUMP=10
      IOFN=1
      IOFO=2
      IOPN=3
      IOPO=4
      IOSC1=11
      IOSC2=12
C   READ CARD 1
      READ(5,1000)(ITITLE(I),I=1,20)
 1000 FORMAT(20A4)
      WRITE(6,1010)(ITITLE(I),I=1,20)
 1010 FORMAT(1H1,10X,20A4)
C   READ CARD 2
      READ(5,1020)NNG,NDG,NTG,NDNSCT,NMAT,IM,JM,KM,IRM,JRM,KRM,NXTP,NYTP
     1,NZTP,NSTEAD,IFLIN,IFLOUT,IGEOM,IETIME
 1020 FORMAT(20I4)
      WRITE(6,1030)NNG,NDG,NTG,NDNSCT,NMAT,IM,JM,KM,IRM,JRM,KRM,NXTP,
     1NYTP,NZTP,NSTEAD,IFLIN,IFLOUT,IGEOM,IETIME
 1030 FORMAT(11X,20I4)
      NPRG=8*IRM*JRM*KRM
      NTOG=NNG+NDG
      IMX=IM-1
      JMX=JM-1
      KMX=KM-1
      NGX=NNG-1
      TIME=1.0D+10
```

```
      IF(IETIME.NE.0)TIME=IETIME
      IMEM=1
      CALL MEMORY(IMEM)
      IF(IMEM.EQ.5) GO TO 999
      IF(IOPT.EQ.0) GO TO 110
      REWIND IOFN
      REWIND IOFD
      REWIND IOPN
      REWIND IOPD
      REWIND IOSC1
      REWIND IOSC2
  110 CALL CALLER
      READ(5,1040) INDIC
      ITEMP4=9999
 1040 FORMAT(I4)
      IF(INDIC.NE.ITEMP4)GO TO 99
      IF(INDIC.EQ.ITEMP4)WRITE(6,1050)
 1050 FORMAT(1H0,10X,'LAST CASE COMPLETED')
  999 STOP
      END
```

```
      SUBROUTINE MEMORY(IMEM)                                           MEM00010
      IMPLICIT REAL*8 (A-H,O-Z)                                         MEM00020
      INTEGER*2 MMAP,NPRMP                                              MEM00030
      COMMON/POINT/LV,LXI,LXIM,LXNU,LSIGF,LSIGR,LSIGT,LSIGS,LALAM,LBETA,MEM00040
     1LXIP,LX,LY,LZ,LHX,LHY,LHZ,LIBP,LJBP,LKBP,LDD1,LDD2,LDD3,LDD4,LDD5,MEM00050
     2LDD6,LDD7,LVO,LMMAP,LNPRMP,LPSI,LP1,LP2,LP3,LFRO,LFRN,LFO,LFN,LSRCMEM00060
     3,LWA,LGA,LSOLN,LOMEG,LXFISS,LXINSC,LXREM,LXLEK,LTOT,LPSO,LW,LPO,LWMEM00070
     4)                                                                 MEM00071
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   MEM00080
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    MEM00090
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,MEM00100
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     MEM00110
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    MEM00120
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    MEM00130
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           MEM00140
      GO TO(100,200),IMEM                                               MEM00150
  100 IOPT=0                                                            MEM00160
      LV=1                                                              MEM00170
      LXI=LV+NNG                                                        MEM00180
      LXIM=LXI+NNG                                                      MEM00190
      LXNU=LXIM+NNG                                                     MEM00200
      LSIGF=LXNU+NMAT*NNG                                               MEM00210
      LSIGR=LSIGF+NMAT*NNG                                              MEM00220
      LSIGT=LSIGR+NMAT*NNG                                              MEM00230
      LSIGS=LSIGT+NMAT*NNG                                              MEM00240
      LALAM=LSIGS+NMAT*NNG*NDNSCT                                       MEM00250
      LBETA=LALAM+NDG                                                   MEM00260
      LXIP=LBETA+NDG                                                    MEM00270
      LX=LXIP+NNG*NDG                                                   MEM00280
      LY=LX+IM                                                          MEM00290
      LZ=LY+JM                                                          MEM00300
      LHX=LZ+KM                                                         MEM00310
      LHY=LHX+IRM                                                       MEM00320
      LHZ=LHY+JRM                                                       MEM00330
      LIBP=LHZ+KRM                                                      MEM00340
      LJBP=LIBP+(IRM+1)/2                                               MEM00350
```

```
      LKBP=LJBP+(JRM+1)/2                                               MEM00360
      LDD1=LKBP+(KRM+1)/2                                               MEM00370
      LDD2=LDD1+NPRG*NNG                                                MEM00380
      LDD3=LDD2+NPRG*NNG                                                MEM00390
      LDD4=LDD3+NPRG*NNG                                                MEM00400
      LDD5=LDD4+NPRG*NNG                                                MEM00410
      LDD6=LDD5+NPRG*NNG                                                MEM00420
      LDD7=LDD6+NPRG*NNG                                                MEM00430
      LVO=LDD7+NPRG*NGX*NDNSCT                                          MEM00431
      LMMAP=LVO+NPRG                                                    MEM00440
      LNPRMP=LMMAP+(IRM*JRM*KRM+3)/4                                    MEM00450
C     NOW COMPUTE THOSE POINTERS WHICH MAY VARY WITH IOPT               MEM00460
  110 LPSI=LNPRMP+(IM*JM*KM+3)/4                                        MEM00470
      LP1=LPSI+(1-IOPT)*IM*JM*KM*NNG+IOPT                               MEM00480
      LP2=LP1+IM*JM*IOPT+(1-IOPT)                                       MEM00490
      LP3=LP2+IM*JM*IOPT+(1-IOPT)                                       MEM00500
      LFRO=LP3+IM*JM*IOPT+(1-IOPT)                                      MEM00510
      LFRN=LFRO+(1-IOPT)*IM*JM*KM+IOPT                                  MEM00520
      LFO=LFRN+(1-IOPT)*IM*JM*KM+IOPT                                   MEM00530
      LFN=LFO+IM*JM*IOPT+(1-IOPT)                                       MEM00540
      LSRC=LFN+IM*JM*IOPT+(1-IOPT)                                      MEM00550
      LWA=LSRC+IM*JM*KM                                                 MEM00560
      LGA=LWA+IM                                                        MEM00570
      LSOLN=LGA+IM                                                      MEM00580
      LOMEG=LSOLN+IM                                                    MEM00590
      LXFISS=LOMEG+NNG                                                  MEM00600
      LXINSC=LXFISS+NNG                                                 MEM00610
      LXREM=LXINSC+NNG                                                  MEM00620
      LXLEK=LXREM+NNG                                                   MEM00630
      LTOT=LXLEK+NNG                                                    MEM00640
      IF(IASIZE-LTOT)120,140,140                                        MEM00650
  120 IOPT=IOPT+1                                                       MEM00660
      IF(IOPT.GT.1)GO TO 130                                            MEM00670
      GO TO 110                                                         MEM00680
  130 IMEM=5                                                            MEM00690
      WRITE(6,1000)IASIZE,LTOT                                          MEM00700
```

```
 1000 FORMAT(1H ,10X,I6,2X,'WORDS ALLOTTED,',2X,I6,2X,'WORDS NEEDED,CORE
     1 CAPACITY EXCEEDED')
      GO TO 300
  140 WRITE(6,1010)IASIZE,LTOT
 1010 FORMAT(1H ,10X,I6,2X,'WORDS ALLOTTED,',2X,I6,2X,'WORDS USED')
      GO TO 300
C     BRANCH TO HERE TO COMPUTE DIMENSION POINTERS THAT CHANGE FOR
C     KINETICS CALCULATION
  200 LP1=LPSI+(1-IOPT)*IM*JM*KM*NTOG+IOPT
      LP2=LP1+IM*JM*NTOG*IOPT+(1-IOPT)
      LP3=LP2+IM*JM*NTOG*IOPT+(1-IOPT)
      LPSO=LP3+IM*JM*NTOG*IOPT+(1-IOPT)
      LW=LPSO+(1-IOPT)*IM*JM*KM+IOPT
      LPO=LW+IM*JM*KM
      LW1=LPO+IM*JM*IOPT+(1-IOPT)
      LTOT=LW1+IM*JM*IOPT+(1-IOPT)
      IF(IASIZE-LTOT)210,230,230
  210 IOPT=IOPT+1
      IF(IOPT.GT.1)GO TO 220
      GO TO 200
  220 IMEM=5
      WRITE(6,1000)IASIZE,LTOT
      GO TO 300
  230 WRITE(6,1010)IASIZE,LTOT
  300 RETURN
      END
```

```
      SUBROUTINE CALLER                                                 CAL00010
      IMPLICIT REAL*8 (A-H,O-Z)                                         CAL00020
      INTEGER*2 MMAP,NPRMP                                              CAL00030
      COMMON/POINT/LV,LXI,LXIM,LXNU,LSIGF,LSIGR,LSIGT,LSIGS,LALAM,LBETA,CAL00040
     1LXIP,LX,LY,LZ,LHX,LHY,LHZ,LIBP,LJBP,LKBP,LDD1,LDD2,LDD3,LDD4,LDD5,CAL00050
     2LDD6,LDD7,LVO,LMMAP,LNPRMP,LPSI,LP1,LP2,LP3,LFRO,LFRN,LFO,LFN,LSRCCAL00060
     3,LWA,LGA,LSOLN,LOMEG,LXFISS,LXINSC,LXREM,LXLEK,LTOT,LPSO,LW,LPO,LWCAL00070
     41                                                                 CAL00071
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   CAL00080
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    CAL00090
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,CAL00100
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     CAL00110
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    CAL00120
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    CAL00130
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           CAL00140
      COMMON/ARAY/A(1)                                                  CAL00150
C   CALL INPUT FOR REMAINDER OF INPUT DATA                              CAL00160
      CALL INPUT(A(LV),A(LXI),A(LXNU),A(LSIGF),A(LSIGR),A(LSIGT),       CAL00170
     1A(LSIGS),A(LALAM),A(LBETA),A(LXIP),A(LX),A(LY),A(LZ),A(LHX),      CAL00180
     2A(LHY),A(LHZ),A(LIBP),A(LJBP),A(LKBP),A(LMMAP),A(LOMEG),NNG,NDG,  CAL00190
     3NDNSCT,NMAT,IM,JM,KM,IRM,JRM,KRM)                                 CAL00200
C   CALL EDIT TO PRINT OUT EDITED VERSION OF PROBLEM DESCRIPTION        CAL00210
      CALL IOEDIT(A(LV),A(LXI),A(LXNU),A(LSIGF),A(LSIGR),A(LSIGT),      CAL00220
     1A(LSIGS),A(LALAM),A(LBETA),A(LXIP),A(LX),A(LY),A(LZ),A(LHX),      CAL00230
     2A(LHY),A(LHZ),A(LIBP),A(LJBP),A(LKBP),A(LMMAP),A(LOMEG),NNG,NDG,  CAL00240
     3NDNSCT,NMAT,IM,JM,KM,IRM,JRM,KRM)                                 CAL00250
C   CALL FLUXIN TO INPUT INITIAL FLUX GUESS                             CAL00260
      CALL FLUXIN(A(LPSI),A(LP1),NNG,IM,JM,KM)                          CAL00270
C   CALL SSTATE TO COMPUTE COEFFICIENTS, SET UP PROBLEM REGIONS, AND    CAL00280
C   COMPUTE STEADY STATE FLUXES(IF REQUESTED)                           CAL00290
      CALL SSTATE(A(LV),A(LXI),A(LXIM),A(LXNU),A(LSIGF),A(LSIGR),       CAL00300
     1A(LSIGT),A(LSIGS),A(LALAM),A(LBETA),A(LXIP),A(LX),A(LY),A(LZ),    CAL00310
     2A(LHX),A(LHY),A(LHZ),A(LIBP),A(LJBP),A(LKBP),A(LDD1),A(LDD2),     CAL00320
     3A(LDD3),A(LDD4),A(LDD5),A(LDD6),A(LDD7),A(LVO),A(LMMAP),A(LNPRMP),CAL00330
     4A(LPSI),A(LP1),A(LP2),A(LP3),A(LFRO),A(LFRN),A(LFO),A(LFN),A(LSRC)CAL00340
     5,A(LWA),A(LGA),A(LSOLN),A(LOMEG),A(LXFISS),A(LXINSC),A(LXREM),A(LXCAL00350
```

```
     6LEK),NNG,NDG,NDNSCT,NMAT,IM,JM,KM,IRM,JRM,KRM,NPRG,NGX)
      IF(ITEMP.NE.4)GO TO 200
      IF(NSTEAD.EQ.2)GO TO 200
      WRITE(6,1000)
 1000 FORMAT(1H1,///,10X,'PROCEEDING INTO TIME-DEPENDENT CALCULATION')
C    CALL FLUXTR TO WRITE FLUXES OUT ON IOSC1 FOR PASSAGE TO TIMDEP
      CALL FLUXTR(A(LPSI),A(LP2),NNG,IM,JM,KM)
C    CALL MEMORY TO REBUILD STORAGE FOR TIME-DEPENDENT CALCULATION
      IMEM=2
      CALL MEMORY(IMEM)
      IF(IMEM.EQ.5)GO TO 200
C    CALL TIMDEP TO PERFORM TIME-DEPENDENT CALCULATION
      CALL TIMDEP(A(LV),A(LXI),A(LXIM),A(LXNU),A(LSIGF),A(LSIGR),
     1A(LSIGT),A(LSIGS),A(LALAM),A(LBETA),A(LXIP),A(LX),A(LY),A(LZ),A(LH
     2X),A(LHY),A(LHZ),A(LIBP),A(LJBP),A(LKBP),A(LDD1),A(LDD2),A(LDD3),
     3A(LDD4),A(LDD5),A(LDD6),A(LDD7),A(LVO),A(LMMAP),A(LNPRMP),A(LPSI)
     4,A(LP1),A(LP2),A(LP3),A(LPSO),A(LW),A(LPD),A(LW1),NNG,NDG,NTOG,
     5NDNSCT,NMAT,IM,JM,KM,IRM,JRM,KRM,NPRG,NGX)
C    NOW RETURN TO MAIN
  200 RETURN
      END
```

```
SUBROUTINE ETIME
IMPLICIT REAL*8 (A-H,O-Z)
INTEGER TNOW,TSTART,TREL,TIO
CALL TIMING(TSTART,TIO)
RETURN
ENTRY ETIMEF(TI)
CALL TIMING(TNOW,TIO)
TREL=TNOW-TSTART
IF(TREL.LT.0)TREL=TREL+8640000
TI=TREL/6000.
RETURN
END
```

```
      SUBROUTINE INPUT(V,XI,XNU,SIGF,SIGR,SIGT,SIGS,ALAM,BETA,XIP,X,Y,Z,
     1HX,HY,HZ,IBP,JBP,KBP,MMAP,OMEG,NNGV,NDGV,NDNSCV,NMATV,IMV,JMV,KMV,
     2IRMV,JRMV,KRMV)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 MMAP,NPRMP
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT
      DIMENSION V(NNGV),XI(NNGV),XNU(NMATV,NNGV),SIGF(NMATV,NNGV),
     1SIGR(NMATV,NNGV),SIGT(NMATV,NNGV),SIGS(NMATV,NNGV,NDNSCV),ALAM(NDG
     2V),BETA(NDGV),XIP(NNGV,NDGV),X(IMV),Y(JMV),Z(KMV),HX(IRMV),
     3HY(JRMV),HZ(KRMV),IBP(IRMV),JBP(JRMV),KBP(KRMV),MMAP(IRMV,JRMV,
     4KRMV),OMEG(NNGV)
      DO 100 I=1,5
      IXTP(I)=0
      IYTP(I)=0
      IZTP(I)=0
  100 CONTINUE
C   READ IN REMAINDER OF TIME-INDEPENDENT INFORMATION
C   ONLY EFFK IS USED IF NSTEAD=0
C     READ CARD 3
      READ(5,1000)EFFK,ORFP,EPS1,EPS2,NOIT,NIIT,NPIT
 1000 FORMAT(D16.10,4X,3D10.4,3I4)
      WRITE(6,1010)EFFK,ORFP,EPS1,EPS2,NOIT,NIIT,NPIT
 1010 FORMAT(11X,D16.10,4X,3D10.4,3I4)
      READ(5,1001)(OMEG(NG),NG=1,NNG)
 1001 FORMAT(8E10.4)
      WRITE(6,1002)(OMEG(NG),NG=1,NNG)
 1002 FORMAT(11X,8E10.4,/(10X,8E10.4))
C     READ CARDS 4
  105 READ(5,1020)NLBC,(IBP(IR),HX(IR),IR=1,IRM)
 1020 FORMAT(I5,5(I5,E10.4)/5(I5,E10.4))
```

```
      WRITE(6,1030)NLBC,(IBP(IR),HX(IR),IR=1,IRM)
 1030 FORMAT(11X,I5,5(I5,E10.4)/(10X,5(I5,E10.4)))
      READ(5,1020)NFBC,(JBP(JR),HY(JR),JR=1,JRM)
      WRITE(6,1030)NFBC,(JBP(JR),HY(JR),JR=1,JRM)
      READ(5,1020)NBBC,(KBP(KR),HZ(KR),KR=1,KRM)
      WRITE(6,1030)NBBC,(KBP(KR),HZ(KR),KR=1,KRM)
C   GENERATE MESH SPACINGS AND MESH PLANE DISTANCES FROM ORIGIN
      IS=1
      ISS=2
      DO 120 IR=1,IRM
      HX(IR)=HX(IR)/(IBP(IR)-IS)
      IS=IBP(IR)
      DO 110 I=ISS,IS
  110 X(I)=X(I-1)+HX(IR)
  120 ISS=IBP(IR)+1
      IS=1
      ISS=2
      DO 140 JR=1,JRM
      HY(JR)=HY(JR)/(JBP(JR)-IS)
      IS=JBP(JR)
      DO 130 J=ISS,IS
  130 Y(J)=Y(J-1)+HY(JR)
  140 ISS=JBP(JR)+1
      IS=1
      ISS=2
      DO 160 KR=1,KRM
      HZ(KR)=HZ(KR)/(KBP(KR)-IS)
      IS=KBP(KR)
      DO 150 K=ISS,IS
  150 Z(K)=Z(K-1)+HZ(KR)
  160 ISS=KBP(KR)+1
C   READ TEST POINTS FOR KINETICS CALCULATIONS CARD 5
      READ(5,1040)(IXTP(I),I=1,NXTP),(IYTP(I),I=1,NYTP),(IZTP(I),I=1,NZT
     1P)
      WRITE(6,1050)(IXTP(I),I=1,NXTP),(IYTP(I),I=1,NYTP),(IZTP(I),I=1,NZ
     1TP)
```

```
C     READ IN MATERIAL REGION MAP CARDS 6
      DO 170 KR=1,KRM
      READ(5,1040)((MMAP(IR,JR,KR),IR=1,IRM),JR=1,JRM)
      WRITE(6,1050)((MMAP(IR,JR,KR),IR=1,IRM),JR=1,JRM)
  170 CONTINUE
 1040 FORMAT(20I4)
 1050 FORMAT(11X,20I4)
C     READ VELOCITIES CARD 7
      READ(5,1060)(V(NG),NG=1,NNG)
      WRITE(6,1070)(V(NG),NG=1,NNG)
 1060 FORMAT(6E12.6)
 1070 FORMAT(11X,6E12.6/(10X,6E12.6))
C     READ FISSION SPECTRUM CARD 8
      READ(5,1060)(XI(NG),NG=1,NNG)
      WRITE(6,1070)(XI(NG),NG=1,NNG)
C     READ MATERIAL PROPERTIES
      DO 190 NM=1,NMAT
C     READ CARD 9
      DO 180 NG=1,NNG
      READ(5,1060)XNU(NM,NG),SIGF(NM,NG),SIGR(NM,NG),SIGT(NM,NG)
  180 WRITE(6,1070)XNU(NM,NG),SIGF(NM,NG),SIGR(NM,NG),SIGT(NM,NG)
C     READ CARD 10
      READ(5,1060)((SIGS(NM,NG,NDNSC),NDNSC=1,NDNSCT),NG=1,NNG)
      WRITE(6,1070)((SIGS(NM,NG,NDNSC),NDNSC=1,NDNSCT),NG=1,NNG)
  190 CONTINUE
C     READ PRECURSOR DATA CARD 11
      DO 200 ND=1,NDG
      READ(5,1060)ALAM(ND),BETA(ND),(XIP(NG,ND),NG=1,NNG)
      WRITE(6,1070)ALAM(ND),BETA(ND),(XIP(NG,ND),NG=1,NNG)
  200 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE IOEDIT(V,XI,XNU,SIGF,SIGR,SIGT,SIGS,ALAM,BETA,XIP,X,Y,Z
     1,HX,HY,HZ,IBP,JBP,KBP,MMAP,OMEG,NNGV,NDGV,NDNSCV,NMATV,IMV,JMV,KMV
     2,IRMV,JRMV,KRMV)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 MMAP,NPRMP
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT
      DIMENSION V(NNGV),XI(NNGV),XNU(NMATV,NNGV),SIGF(NMATV,NNGV),
     1SIGR(NMATV,NNGV),SIGT(NMATV,NNGV),SIGS(NMATV,NNGV,NDNSCV),ALAM(NDG
     2V),BETA(NDGV),XIP(NNGV,NDGV),X(IMV),Y(JMV),Z(KMV),HX(IRMV),
     3HY(JRMV),HZ(KRMV),IBP(IRMV),JBP(JRMV),KBP(KRMV),MMAP(IRMV,JRMV,
     4KRMV),OMEG(NNGV)
      WRITE(6,1000)(ITITLE(I),I=1,20)
 1000 FORMAT(1H1,10X,'3DKIN RUN FOR',2X,20A4)
C   WILL ADD REST OF EDITING ROUTINE LATER
      RETURN
      END
```

```
      SUBROUTINE FLUXIN(PSI,PI,NNGV,IMV,JMV,KMV)                          FLU00010
      IMPLICIT REAL*8 (A-H,O-Z)                                           FLU00020
      INTEGER*2 MMAP,NPRMP                                                FLU00030
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,    FLU00080
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),     FLU00090
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,FLU00100
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,      FLU00110
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    FLU00120
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,     FLU00130
     1TEMP5,TEMP5,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT            FLU00140
      DIMENSION PSI(NNGV,IMV,JMV,KMV),PI(IMV,JMV)                        FLU00150
      ITEMP=IFLIN+1                                                      FLU00160
      ITEMP1=IOPT+1                                                      FLU00170
      PI=3.14159265358979D0                                              FLU00180
      TWO=2.0D0                                                          FLU00190
      GO TO(100,300,400),ITEMP                                           FLU00200
C   BRANCH HERE FOR SINE FLUX GUESS                                      FLU00210
  100 DO 200 NG=1,NNG                                                    FLU00220
      DO 200 K=1,KM                                                      FLU00230
      IF(NBBC.EQ.1)GO TO 110                                             FLU00240
      TEMP1=DSIN((K-1)*PI/(KM-1))                                        FLU00250
      GO TO 120                                                          FLU00260
  110 TEMP1=DCOS((K-1)*PI/(TWO*(KM-1)))                                  FLU00270
  120 IF(K.EQ.KM)TEMP1=0.0D0                                             FLU00280
      DO 190 J=1,JM                                                      FLU00290
      IF(NFBC.EQ.1)GO TO 130                                             FLU00300
      TEMP2=DSIN((J-1)*PI/(JM-1))                                        FLU00310
      GO TO 140                                                          FLU00320
  130 TEMP2=DCOS((J-1)*PI/(TWO*(JM-1)))                                  FLU00330
  140 IF(J.EQ.JM)TEMP2=0.0D0                                             FLU00340
      DO 180 I=1,IM                                                      FLU00350
      IF(NLBC.EQ.1)GO TO 150                                             FLU00360
      TEMP3=DSIN((I-1)*PI/(IM-1))                                        FLU00370
      GO TO 160                                                          FLU00380
  150 TEMP3=DCOS((I-1)*PI/(TWO*(IM-1)))                                  FLU00390
  160 IF(I.EQ.IM)TEMP3=0.0D0                                             FLU00400
```

```
      IF(ITEMP1.EQ.2)GO TO 170
      PSI(NG,I,J,K)=TEMP1*TEMP2*TEMP3
      GO TO 180
  170 P1(I,J)=TEMP1*TEMP2*TEMP3
  180 CONTINUE
  190 CONTINUE
      IF(ITEMP1.EQ.1)GO TO 200
      WRITE(IOPN) P1
  200 CONTINUE
      GO TO 999
C     BRANCH HERE FOR FLUXES INPUT ON CARDS
  300 DO 340 NG=1,NNG
      DO 340 K=1,KM
      GO TO(310,320),ITEMP1
  310 READ(5,1000)((PSI(NG,I,J,K),I=1,IM),J=1,JM)
      IF(K.LT.KM)GO TO 330
      DO 315 J=1,JM
      DO 315 I=1,IM
  315 PSI(NG,I,J,KM)=0.0D0
      GO TO 330
  320 READ(5,1000)((P1(I,J),I=1,IM),J=1,JM)
      IF(K.LT.KM)GO TO 325
      DO 324 J=1,JM
      DO 324 I=1,IM
  324 P1(I,J)=0.0D0
  325 WRITE(IOPN)P1
  330 CONTINUE
  340 CONTINUE
 1000 FORMAT(5D16.10)
      GO TO 999
C     BRANCH HERE FOR FLUXES INPUT ON TAPE
  400 DO 440 NG=1,NNG
      DO 440 K=1,KM
      GO TO(410,420),ITEMP1
  410 READ(IOPSI)((PSI(NG,I,J,K),I=1,IM),J=1,JM)
      IF(K.LT.KM)GO TO 430
```

```
      DO 415 J=1,JM
      DO 415 I=1,IM
  415 PSI(NG,I,J,KM)=0.0D0
      GO TO 430
  420 READ(IOPSI)P1
      IF(K.LT.KM)GO TO 425
      DO 424 J=1,JM
      DO 424 I=1,IM
  424 P1(I,J)=0.0D0
  425 WRITE(IOPN)P1
  430 CONTINUE
  440 CONTINUE
  999 IF(ITEMP1.EQ.2)REWIND IOPN
      RETURN
      END
```

```
      SUBROUTINE SSTATE(V,XI,XIM,XNU,SIGF,SIGR,SIGT,SIGS,ALAM,BETA,XIP,XSST00010
     1,Y,Z,HX,HY,HZ,IBP,JBP,KBP,DD1,DD2,DD3,DD4,DD5,DD6,DD7,VD,MMAP,NPRMSST00020
     2P,PSI,P1,P2,P3,FRO,FRN,FO,FN,SRC,WA,GA,SOLN,OMEG,XFISS,XINSC,XREM,SST00030
     3XLEK,NNGV,NDGV,NDNSCV,NMATV,IMV,JMV,KMV,IRMV,JRMV,KRMV,NPRGV,NGXV)SST00040
      IMPLICIT REAL*8 (A-H,O-Z)                                         SST00050
      INTEGER*2 MMAP,NPRMP                                              SST00060
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   SST00110
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    SST00120
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,SST00130
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     SST00140
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    SST00150
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    SST00160
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           SST00170
      DIMENSION V(NNGV),XI(NNGV),XIM(NNGV),XNU(NMATV,NNGV),             SST00180
     1SIGF(NMATV,NNGV),SIGR(NMATV,NNGV),SIGT(NMATV,NNGV),SIGS(NMATV,NNGVSST00190
     2,NDNSCV),ALAM(NDGV),BETA(NDGV),XIP(NNGV,NDGV),X(IMV),Y(JMV),Z(KMV)SST00200
     3,HX(IRMV),HY(JRMV),HZ(KRMV),IBP(IRMV),JBP(JRMV),KBP(KRMV),DD1(NPRGSST00210
     4V,NNGV),DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),DD5(NPRGV,SST00220
     5NNGV),DD6(NPRGV,NNGV),DD7(NPRGV,NGXV,NDNSCV),MMAP(IRMV,JRMV,KRMV),SST00230
     6NPRMP(IMV,JMV,KMV),PSI(NNGV,IMV,JMV,KMV),P1(IMV,JMV),P2(IMV,JMV), SST00240
     7P3(IMV,JMV),FRO(IMV,JMV,KMV),FRN(IMV,JMV,KMV),FO(IMV,JMV),FN(IMV,JSST00250
     8MV),SRC(IMV,JMV,KMV),WA(IMV),GA(IMV),SOLN(IMV),OMEG(NNGV),XFISS(NNSST00260
     9GV),XINSC(NNGV),XREM(NNGV),XLEK(NNGV),VD(NPRGV)                   SST00270
      WRITE(6,1000)(ITITLE(I),I=1,20)                                   SST00280
 1000 FORMAT(1H1,10X,'SSTATE ENTERED FOR',2X,20A4)                      SST00290
C    CALL SETUP1 TO COMPUTE PROBLEM REGION NUMBERS, GENERATE NPRMP(I,J,K)SST00300
C    AND COMPUTE COEFFICIENTS                                           SST00310
      CALL SETUP1(V,XI,XNU,SIGF,SIGR,SIGT,SIGS,X,Y,Z,HX,HY,HZ,IBP,JBP,  SST00320
     1KBP,DD1,DD2,DD3,DD4,DD5,DD6,DD7,VD,MMAP,NPRMP,NNG,NDG,NDNSCT,NMAT,SST00330
     2IM,JM,KM,IRM,JRM,KRM,NPRG,NGX)                                    SST00340
C    SWITCH FLUX TAPE DESIGNATIONS                                      SST00350
      ITEMP=IOPO                                                        SST00360
      IOPO=IOPN                                                         SST00370
      IOPN=ITEMP                                                        SST00380
      ITEMP=4                                                           SST00390
      ONE=1.0D0                                                         SST00392
```

```
      HALF=0.5D0                                                        SST00393
      BETAT=0.0D0                                                       SST00394
      DO 80 ND=1,NDG                                                    SST00395
   80 BETAT=BETAT+BETA(ND)                                              SST00396
      DO 85 NG=1,NNG                                                    SST00400
   85 XIM(NG)=XI(NG)*(1.0D0-BETAT)/EFFK                                 SST00410
      IF(NSTEAD.EQ.0) GO TO 540                                         SST00420
      DO 90 NG=1,NNG                                                    SST00430
      IF(OMEG(NG).LT..95.OR.OMEG(NG).GT.1.05)GO TO 90                   SST00440
      GO TO 95                                                          SST00450
   90 CONTINUE                                                          SST00460
      GO TO 99                                                          SST00470
   95 CALL ORPEST(X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5,MMAP,NPRMP,PSI,    SST00480
     1P1,P2,P3,FRO,FRN,FO,FN,SRC,WA,GA,SOLN,OMEG,XFISS,XINSC,XREM,      SST00490
     2XLEK,NNG,NMAT,IM,JM,KM,IRM,JRM,KRM,NPRG)                          SST00500
C    COMPUTE POINT FISSION SOURCE                                       SST00510
   99 DO 140 NG=1,NNG                                                   SST00520
      XFISS(NG)=0.0D0                                                   SST00530
      VOLB=ONE                                                          SST00531
      VOLF=ONE                                                          SST00532
      VOLL=ONE                                                          SST00533
      IF(NBBC.EQ.1)VOLB=HALF                                            SST00534
      DO 140 K=1,KM                                                     SST00540
      IF(K.GT.1)VOLB=ONE                                                SST00541
      IF(NFBC.EQ.1)VOLF=HALF                                            SST00542
      IF(IOPT.EQ.0) GO TO 100                                           SST00550
      READ (IDPO)P2                                                     SST00560
      IF(K.EQ.KM) GO TO 140                                             SST00570
  100 DO 130 J=1,JMX                                                    SST00580
      IF(J.GT.1)VOLF=ONE                                                SST00581
      VOLC=VOLF*VOLB                                                    SST00582
      IF(NLBC.EQ.1)VOLL=HALF                                            SST00583
      DO 130 I=1,IMX                                                    SST00590
      IF(I.GT.1)VOLL=ONE                                                SST00591
      VOLD=VOLL*VOLC                                                    SST00592
      NPR=NPRMP(I,J,K)                                                  SST00600
```

```
      IF(IOPT.EQ.1) GO TO 110                                           SST00610
      FRO(I,J,K)=FRO(I,J,K)+DD6(NPR,NG)*PSI(NG,I,J,K)                   SST00620
      XFISS(NG)=XFISS(NG)+DD6(NPR,NG)*PSI(NG,I,J,K)*VOLD                SST00630
      GO TO 120                                                         SST00640
C   IF FISSION SOURCE ON I/O, STORE TEMPORARILY IN SRC(I,J,K)           SST00650
  110 SRC(I,J,K)=SRC(I,J,K)+DD6(NPR,NG)*P2(I,J)                         SST00660
      XFISS(NG)=XFISS(NG)+DD6(NPR,NG)*P2(I,J)*VOLD                      SST00670
  120 CONTINUE                                                          SST00680
  130 CONTINUE                                                          SST00690
  140 CONTINUE                                                          SST00700
      XFISST=0.0D0                                                      SST00710
      TEMP=0.0D0                                                        SST00720
      IF(EFFK.LT.0.1.OR.EFFK.GT.10.0)EFFK=1.0D0                         SST00730
      ALAMN=ONE                                                         SST00740
      DO 150 NG=1,NNG                                                   SST00750
  150 TEMP=TEMP+XFISS(NG)                                               SST00760
      DO 160 NG=1,NNG                                                   SST00770
      TEMP2=0.0D0                                                       SST00772
      DO 155 ND=1,NDG                                                   SST00773
  155 TEMP2=TEMP2+XIP(NG,ND)*BETA(ND)/EFFK                              SST00775
      XIM(NG)=XI(NG)*(1.0D0-BETAT)/EFFK                                 SST00780
      XFISS(NG)=(XIM(NG)+TEMP2)*TEMP                                    SST00790
  160 XFISST=XFISST+XFISS(NG)                                           SST00800
      IF(IOPT.EQ.0) GO TO 180                                           SST00810
      DO 170 K=1,KM                                                     SST00820
  170 WRITE (IOFO)((SRC(I,J,K),I=1,IM),J=1,JM)                          SST00830
      REWIND IOFO                                                       SST00840
      REWIND IOPO                                                       SST00850
C   OUTER ITERATION LOOP STARTS HERE                                    SST00860
  180 NOITT=0                                                           SST00870
  190 CONTINUE                                                          SST00880
      NTIT=0                                                            SST00890
      NG=1                                                              SST00900
      FLXCON=0.0D0                                                      SST00910
  200 CONTINUE                                                          SST00920
C   ZERO SOURCE AND ADD IN FISSION SOURCE                               SST00930
```

```
      TEMP=0.0D0                                                        SST00931
      DO 205 ND=1,NDG                                                   SST00932
  205 TEMP=TEMP+XIP(NG,ND)*BETA(ND)/EFFK                                SST00933
      DO 240 K=1,KM                                                     SST00940
      IF(IOPT.EQ.0) GO TO 210                                           SST00950
      READ(IOFO) FO                                                     SST00960
      IF(K.EQ.KM) GO TO 240                                             SST00970
  210 DO 230 J=1,JMX                                                    SST00980
      DO 230 I=1,IMX                                                    SST00990
      SRC(I,J,K)=0.0D0                                                  SST01000
      IF(IOPT.EQ.0) GO TO 220                                           SST01010
      SRC(I,J,K)=SRC(I,J,K)+(XIM(NG)+TEMP)*FO(I,J)                      SST01020
      GO TO 230                                                         SST01030
  220 SRC(I,J,K)=SRC(I,J,K)+(XIM(NG)+TEMP)*FRO(I,J,K)                   SST01040
      IF(NG.EQ.1)FRN(I,J,K)=0.0D0                                       SST01050
  230 CONTINUE                                                          SST01060
  240 CONTINUE                                                          SST01070
      IF(IOPT.EQ.1) REWIND IOFO                                         SST01080
C   ADD IN SCATTERING SOURCES                                           SST01090
      ITEMP1=NG-NDNSCT                                                  SST01100
      IF(ITEMP1.GE.1) GO TO 250                                         SST01110
      ITEMP1=1                                                          SST01120
  250 ITEMP2=NG-1                                                       SST01130
      IF(ITEMP2.LE.NDNSCT)GO TO 260                                     SST01140
      ITEMP2=NDNSCT                                                     SST01150
  260 IF(ITEMP1.GE.NG) GO TO 310                                        SST01160
C   SCATTERING SOURCE TO GROUP NG FROM GROUP ITEMP1                     SST01170
  270 DO 300 K=1,KM                                                     SST01180
      IF(IOPT.EQ.1) READ(IOPN)P2                                        SST01190
      IF(K.EQ.KM)GO TO 300                                              SST01200
      DO 290 J=1,JMX                                                    SST01210
      DO 290 I=1,IMX                                                    SST01220
      NPR=NPRMP(I,J,K)                                                  SST01230
      IF(IOPT.EQ.1) GO TO 280                                           SST01240
      SRC(I,J,K)=SRC(I,J,K)+DD7(NPR,ITEMP1,ITEMP2)*PSI(ITEMP1,I,J,K)    SST01250
      GO TO 290                                                         SST01260
```

```
  280 SRC(I,J,K)=SRC(I,J,K)+DD7(NPR,ITEMP1,ITEMP2)*P2(I,J)              SST01270
  290 CONTINUE                                                          SST01280
  300 CONTINUE                                                          SST01290
  310 ITEMP1=ITEMP1+1                                                   SST01300
      ITEMP2=ITEMP2-1                                                   SST01310
      IF(ITEMP1.LT.NG)GO TO 270                                         SST01320
C   SOURCE NOW CALCULATED  I/O DEVICE IOPO READY TO READ IN FIRST PLANE SST01330
C   FOR GROUP NG IF IOPT=1                                              SST01340
      TEMP=0.0D0                                                        SST01350
      VOLB=ONE                                                          SST01351
      VOLF=ONE                                                          SST01352
      VOLL=ONE                                                          SST01353
      IF(NBBC.EQ.1)VOLB=HALF                                            SST01354
      DO 320 K=1,KMX                                                    SST01360
      IF(K.GT.1)VOLB=ONE                                                SST01361
      IF(NFBC.EQ.1)VOLF=HALF                                            SST01362
      DO 320 J=1,JMX                                                    SST01370
      IF(J.GT.1)VOLF=ONE                                                SST01371
      VOLC=VOLB*VOLF                                                    SST01372
      IF(NLBC.EQ.1)VOLL=HALF                                            SST01373
      DO 320 I=1,IMX                                                    SST01380
      IF(I.GT.1)VOLL=ONE                                                SST01381
      TEMP=TEMP+SRC(I,J,K)*VOLC*VOLL                                    SST01390
  320 CONTINUE                                                          SST01400
      XINSC(NG)=TEMP-XFISS(NG)                                          SST01410
C   NOW PERFORM INNER ITERATIONS FOR GROUP NG                           SST01420
      ITEMP5=1                                                          SST01430
      IF(NOITT.GT.0.AND.FLCOND.LT.1.0D-5)ITEMP5=5                       SST01435
      IF(IOPT.EQ.1) GO TO 330                                           SST01440
      CALL INNER0(X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5,MMAP,NPRMP,PSI,P1, SST01450
     1P2,P3,FO,SRC,WA,GA,SOLN,OMEG,XFISS,XINSC,XREM,XLEK,NNG,NMAT,IM,   SST01460
     2JM,KM,IRM,JRM,KRM,NPRG,NG)                                        SST01470
      IF(TEMP3.GT.FLXCON)FLXCON=TEMP3                                   SST01480
      GO TO 400                                                         SST01490
  330 CALL INNER1(X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5,MMAP,NPRMP,PSI,P1, SST01500
     1P2,P3,FO,SRC,WA,GA,SOLN,OMEG,XFISS,XINSC,XREM,XLEK,NNG,NMAT,IM,   SST01510
```

```
     2JM,KM,IRM,JRM,KRM,NPRG,NG)
      IF(TEMP3.GT.FLXCON)FLXCON=TEMP3
      REWIND IOSC1
      DO 340 ITEMP4=1,NDNSCT
      DO 340 K=1,KM
      BACKSPACE IOPN
  340 CONTINUE
C   IOPN HAS NOW BEEN POSITIONED TO COMPUTE SCATTERING SOURCE FOR NEXT
C   GROUP.  IOSC1 CAN BE USED TO OBTAIN FLUXES FOR COMPUTING FN
      DO 380 K=1,KM
      READ(IOSC1)P2
      IF(K.EQ.KM)GO TO 380
      IF(NG.GT.1)GO TO 360
      DO 350 J=1,JMX
      DO 350 I=1,IMX
      NPR=NPRMP(I,J,K)
  350 SRC(I,J,K)=DD6(NPR,NG)*P2(I,J)
      GO TO 380
  360 READ(IOFN)FN
      DO 370 J=1,JMX
      DO 370 I=1,IMX
      NPR=NPRMP(I,J,K)
  370 SRC(I,J,K)=FN(I,J)+DD6(NPR,NG)*P2(I,J)
  380 CONTINUE
      IF(NG.GT.1)REWIND IOFN
      DO 390 K=1,KM
      WRITE(IOFN)((SRC(I,J,K),I=1,IM),J=1,JM)
  390 CONTINUE
      REWIND IOSC1
      REWIND IOFN
      GO TO 420
  400 DO 410 K=1,KMX
      DO 410 J=1,JMX
      DO 410 I=1,IMX
      NPR=NPRMP(I,J,K)
  410 FRN(I,J,K)=FRN(I,J,K)+DD6(NPR,NG)*PSI(NG,I,J,K)
```

```
  420 NG=NG+1                                                          SST01880
      IF(NG.LE.NNG) GO TO 200                                          SST01890
C   NOW ONE OUTER ITERATION HAS BEEN CONPLETED                         SST01900
C   NEW FISSION SOURCE IS STORED IN SRC IF IOPT=1                      SST01910
      FLCONO=FLXCON                                                    SST01920
      XFISSO=XFISST                                                    SST01930
      TEMP5=0.0D0                                                      SST01940
      TEMP6=0.0D0                                                      SST01950
      VOLB=ONE                                                         SST01951
      VOLF=ONE                                                         SST01952
      VOLL=ONE                                                         SST01953
      IF(NBBC.EQ.1)VOLB=HALF                                           SST01954
      IF(IOPT.EQ.1) GO TO 450                                          SST01960
      DO 430 K=1,KMX                                                   SST01970
      IF(K.GT.1)VOLB=ONE                                               SST01971
      IF(NFBC.EQ.1)VOLF=HALF                                           SST01972
      DO 430 J=1,JMX                                                   SST01980
      IF(J.GT.1)VOLF=ONE                                               SST01981
      VOLC=VOLB*VOLF                                                   SST01982
      IF(NLBC.EQ.1)VOLL=HALF                                           SST01983
      DO 430 I=1,IMX                                                   SST01990
      IF(I.GT.1)VOLL=ONE                                               SST01991
      VOLD=VOLC*VOLL                                                   SST01992
      TEMP5=TEMP5+FRN(I,J,K)*VOLD                                      SST02000
      FRN(I,J,K)=FRO(I,J,K)+ORFP*(FRN(I,J,K)-FRO(I,J,K))               SST02010
  430 TEMP6=TEMP6+FRN(I,J,K)*VOLD                                      SST02020
      TEMP=TEMP5/TEMP6                                                 SST02030
      DO 440 K=1,KMX                                                   SST02040
      DO 440 J=1,JMX                                                   SST02050
      DO 440 I=1,IMX                                                   SST02060
  440 FRO(I,J,K)=TEMP*FRN(I,J,K)                                       SST02070
      GO TO 490                                                        SST02080
  450 DO 460 K=1,KMX                                                   SST02090
      IF(K.GT.1)VOLB=ONE                                               SST02091
      IF(NFBC.EQ.1)VOLF=HALF                                           SST02092
      READ(IOFO) FO                                                    SST02100
```

```
      DO 460 J=1,JMX                                                    SST02110
      IF(J.GT.1)VOLF=ONE                                                SST02111
      VOLC=VOLB*VOLF                                                    SST02112
      IF(NLBC.EQ.1)VOLL=HALF                                            SST02113
      DO 460 I=1,IMX                                                    SST02120
      IF(I.GT.1)VOLL=ONE                                                SST02121
      VOLD=VOLC*VOLL                                                    SST02122
      TEMP5=TEMP5+SRC(I,J,K)*VOLD                                       SST02130
      SRC(I,J,K)=FO(I,J)+ORFP*(SRC(I,J,K)-FO(I,J))                      SST02140
  460 TEMP6=TEMP6+SRC(I,J,K)*VOLD                                       SST02150
      TEMP=TEMP5/TEMP6                                                  SST02160
      DO 480 K=1,KMX                                                    SST02170
      DO 470 J=1,JMX                                                    SST02180
      DO 470 I=1,IMX                                                    SST02190
  470 FN(I,J)=SRC(I,J,K)*TEMP                                           SST02200
      WRITE(IOFN)FN                                                     SST02210
  480 CONTINUE                                                          SST02220
      REWIND IOFO                                                       SST02230
      REWIND IOFN                                                       SST02240
      REWIND IOPO                                                       SST02250
      REWIND IOPN                                                       SST02260
  490 XFISST=0.0D0                                                      SST02270
      DO 500 NG=1,NNG                                                   SST02280
      TEMP1=0.0D0                                                       SST02281
      DO 495 ND=1,NDG                                                   SST02282
  495 TEMP1=TEMP1+BETA(ND)*XIP(NG,ND)/EFFK                              SST02283
      XFISS(NG)=(XIM(NG)+TEMP1)*TEMP5                                   SST02290
  500 XFISST=XFISST+XFISS(NG)                                           SST02300
      ALAMO=ALAMN                                                       SST02310
      ALAMN=XFISST/XFISSO                                               SST02320
      DO 510 NG=1,NNG                                                   SST02330
      XFISS(NG)=XFISS(NG)/ALAMN                                         SST02340
  510 XIM(NG)=XIM(NG)/ALAMN                                             SST02350
      XFISST=XFISST/ALAMN                                               SST02360
C     CONVERGENCE TESTS                                                 SST02370
      NGOTO=1                                                           SST02380
```

```
      NOITT=NOITT+1
      IF(IETIME.EQ.0)GO TO 520
      CALL ETIMEF(TEMP)
      IF(TEMP.GT.TIME)NGOTO=2
  520 IF(NOITT.GE.NOIT)NGOTO=3
      IF(DABS(1.0D0-ALAMN).LE.EPS1.AND.FLXCON.LE.EPS2)NGOTO=4
C   COMPUTE NEW K-EFFECTIVE
      EFFK=0.0D0
      TEMP=0.0D0
      DO 530 NG=1,NNG
      EFFK=EFFK+XIM(NG)
  530 TEMP=TEMP+XI(NG)
      EFFK=(TEMP/EFFK)*(1.0D0-BETAT)
C   SWITCH I/O DEVICES
      ITEMP1=IOPN
      IOPN=IOPO
      IOPO=ITEMP1
      ITEMP1=IOFN
      IOFN=IOFO
      IOFO=ITEMP1
C   IF IOPT=1, LATEST FLUXES ON IOPO AND LATEST FISSION SOURCE ON IOFO
C   CALL STEADY STATE ITERATION PRINT MONITOR
      CALL SSTOUT(PSI,P2,NNG,IM,JM,KM,NGOTO,NOITT)
C   IF NGOTO=1, LOOP TO 190 TO BEGIN ANOTHER OUTER ITERATION
C   IF NGOTO=2, HAVE EXCEEDED RUNNING TIME
C   IF NGOTO=3, HAVE REACHED MAX. NO. OF OUTER ITERATIONS
C   IF NGOTO=4, HAVE ACHIEVED CONVERGENCE, CAN GO ON TO TIME-DEP CALC.
      ITEMP=NGOTO
      GO TO (190,540,540,540),NGOTO
  540 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ORPEST(X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5,MMAP,NPRMP,
     1PSI,P1,P2,P3,FRO,FRN,FO,FN,SRC,WA,GA,SOLN,OMEG,XFISS,XINSC,XREM,
     2XLEK,NNGV,NMATV,IMV,JMV,KMV,IRMV,JRMV,KRMV,NPRGV)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 MMAP,NPRMP
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT
      DIMENSION X(IMV),Y(JMV),Z(KMV),HX(IRMV),HY(JRMV),HZ(KRMV),
     1DD1(NPRGV,NNGV),DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),
     2DD5(NPRGV,NNGV),MMAP(IRMV,JRMV,KRMV),NPRMP(IMV,JMV,KMV),
     3PSI(NNGV,IMV,JMV,KMV),P1(IMV,JMV),P2(IMV,JMV),P3(IMV,JMV),
     4FRO(IMV,JMV,KMV),FRN(IMV,JMV,KMV),FO(IMV,JMV),FN(IMV,JMV),
     5SRC(IMV,JMV,KMV),WA(IMV),GA(IMV),SOLN(IMV),OMEG(NNGV),XFISS(NNGV),
     6XINSC(NNGV),XREM(NNGV),XLEK(NNGV)
C   SAVE NIIT AND NPIT
      ITEMP1=NIIT
      ITEMP2=NPIT
      ITEMP5=5
C   INITIALIZE SRC
      DO 100 K=1,KM
      DO 100 J=1,JM
      DO 100 I=1,IM
  100 SRC(I,J,K)=0.0D0
      DO 260 NG=1,NNG
C   STORE INITIAL FLUXES FOR GROUP NG IN FRO IF IOPT=0
      IF(IOPT.EQ.1)GO TO 120
      DO 110 K=1,KM
      DO 110 J=1,JM
      DO 110 I=1,IM
  110 FRO(I,J,K)=PSI(NG,I,J,K)
      GO TO 140
```

```
  120 REWIND IOSC1
      DO 130 K=1,KM
      READ(IOPO)P2
      WRITE(IOSC1)P2
  130 CONTINUE
C   NOW INITIALIZE SOME PARAMETERS
  140 NPIT=1
      NIIT=5
      OMEGBJ=0.0D0
      OMEGBL=0.0D0
      ICT=0
      ALAMES=0.0D0
  150 CONTINUE
      IF(IOPT.EQ.1)GO TO 170
      DO 160 K=1,KM
      DO 160 J=1,JM
      DO 160 I=1,IM
  160 FRN(I,J,K)=PSI(NG,I,J,K)
      CALL INNERO(X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5,MMAP,NPRMP,PSI,
     1P1,P2,P3,FD,SRC,WA,GA,SOLN,OMEG,XFISS,XINSC,XREM,XLEK,NNG,NMAT,
     2IM,JM,KM,IRM,JRM,KRM,NPRG,NG)
      GO TO 180
  170 CALL INNER1(X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5,MMAP,NPRMP,PSI,
     1P1,P2,P3,FD,SRC,WA,GA,SOLN,OMEG,XFISS,XINSC,XREM,XLEK,NNG,NMAT,
     2IM,JM,KM,IRM,JRM,KRM,NPRG,NG)
  180 NIIT=1
      ICT=ICT+1
      IF(ICT.LE.1)GO TO 150
C   COMPUTE LAMBDA(M)
      TEMP5=0.0D0
      TEMP6=0.0D0
      IF(IOPT.EQ.1)GO TO 200
      DO 190 K=1,KM
      DO 190 J=1,JM
      DO 190 I=1,IM
      TEMP5=TEMP5+PSI(NG,I,J,K)*PSI(NG,I,J,K)
```

```
      TEMP6=TEMP5+PSI(NG,I,J,K)*FRN(I,J,K)
  190 CONTINUE
      GO TO 230
  200 REWIND IOSC2
      REWIND IOSC1
      DO 220 K=1,KMX
      READ(IOSC1)P2
      READ(IOSC2)P1
      DO 210 J=1,JM
      DO 210 I=1,IM
      TEMP5=TEMP5+P2(I,J)*P2(I,J)
  210 TEMP6=TEMP6+P1(I,J)*P2(I,J)
  220 CONTINUE
  230 ALAMES=TEMP5/TEMP5
      TEMP4=DABS(1.0D0-1.0D0/TEMP4)
      TEMP1=DABS(1.0D0-1.0D0/TEMP1)
      ALAMES=DABS(1.0D0-ALAMES)
      OMEGBU=2.0D0/(1.0D0+DSQRT(TEMP4))
      OMEGBL=2.0D0/(1.0D0+DSQRT(TEMP1))
      OMEGM=2.0D0/(1.0D0+DSQRT(ALAMES))
      IF(DABS(OMEGBU-OMEGBL).LE.((2.0D0-OMEGM)/1.0D1))GO TO 240
      IF(ICT.LT.15)GO TO 150
C   NOW STORE OMEGM AS OMEG(NG)
  240 OMEG(NG)=OMEGM
C   STORE INITIAL FLUXES BACK INTO PSI IF IOPT=0
      IF(IOPT.EQ.1)GO TO 260
      DO 250 K=1,KM
      DO 250 J=1,JM
      DO 250 I=1,IM
      PSI(NG,I,J,K)=FRO(I,J,K)
  250 FRO(I,J,K)=0.0D0
  260 CONTINUE
      IF(IOPT.EQ.1)REWIND IOPO
      WRITE(6,1000)(OMEG(NG),NG=1,NNG)
 1000 FORMAT(1H0,10X,'OPTIMUM OMEGAS NOW COMPUTED'//(10X,6E15.8))
      NIIT=ITEMP1
```

```
      NPIT=ITEMP2                                                       ORP01130
      RETURN                                                            ORP01140
      END                                                               ORP01150
```

```
      SUBROUTINE SSTOUT(PSI,P2,NNGV,IMV,JMV,KMV,NGOTO,NOITT)            SST00010
      IMPLICIT REAL*8 (A-H,O-Z)                                         SST00020
      INTEGER*2 MMAP,NPRMP                                              SST00030
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   SST00080
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    SST00090
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,SST00100
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     SST00110
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    SST00120
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    SST00130
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           SST00140
      DIMENSION PSI(NNGV,IMV,JMV,KMV),P2(IMV,JMV)                       SST00150
      TEMP1=DABS(1.0D0-ALAMO/ALAMN)                                     SST00160
      CALL ETIMEF(TEMP)                                                 SST00170
      IF(NOITT.GT.1)GO TO 100                                           SST00180
      WRITE(6,1010)                                                     SST00190
 1010 FORMAT(1H0,//,53X,'OUTER ITERATION SUMMARY',/)                    SST00200
      WRITE(6,1020)                                                     SST00210
 1020 FORMAT(1H ,11X,'OUTER IT.',5X,'NO. OF INNER',6X,'TOTAL COMP.',7X, SST00220
     1'REL. FLUX',9X,'LAMBDA',27X,'ESTIMATED')                          SST00230
      WRITE(6,1030)                                                     SST00240
 1030 FORMAT(1H ,12X,'NUMBER',9X,'ITERATIONS',7X,'TIME(MIN.)',6X,' CONVESST00250
     1RGENCE',6X,'CONVERGENCE',8X,'LAMBDA',9X,'K-EFFECTIVE',/)          SST00260
  100 WRITE(6,1040)NOITT,NTIT,TEMP,FLXCON,TEMP1,ALAMN,EFFK              SST00270
 1040 FORMAT(1H ,13X,I4,13X,I4,11X,F8.3,6X,3D17.9,1X,F16.12)            SST00280
      IF(NGOTO.EQ.1)GO TO 220                                           SST00290
      WRITE(6,1050)                                                     SST00300
 1050 FORMAT(1H0,10X,'STEADY STATE ITERATIONS TERMINATED')              SST00310
      IF(NGOTO.EQ.2)WRITE(6,1060)                                       SST00320
 1060 FORMAT(1H ,15X,'INSUFFICIENT TIME REMAINING FOR ANOTHER ITERATION'SST00330
     1)                                                                 SST00340
      IF(NGOTO.EQ.3)WRITE(6,1070)                                       SST00350
 1070 FORMAT(1H ,15X,'MAXIMUM NUMBER OF OUTER ITERATIONS EXCEEDED')     SST00360
      IF(NGOTO.EQ.4)WRITE(6,1080)                                       SST00370
 1080 FORMAT(1H ,15X,'CONVERGENCE HAS BEEN ACHIEVED')                   SST00380
C   IF IFLOUT = 0, RETURN                                               SST00390
      IF(IFLOUT.EQ.0)GO TO 220                                          SST00400
```

```
C   IF IFLOUT = 1, 2, OR 3, PRINT FLUXES                                SST00410
      IF(IFLOUT.EQ.4)GO TO 180                                          SST00420
      WRITE(6,1090)(ITITLE(I),I=1,20)                                   SST00430
 1090 FORMAT(1H1,///,10X,'FINAL FLUXES FOR THE RUN  ',20A4)             SST00440
      JME=JM                                                            SST00450
      IF(JM.GT.50)JME=50                                                SST00460
      ITEMP2=50/JME                                                     SST00470
      JMS=1                                                             SST00480
      DO 170 NG=1,NNG                                                   SST00490
      DO 160 K=1,KM                                                     SST00500
      IF(K.GT.1.OR.NG.GT.1)WRITE(6,1100)                                SST00510
 1100 FORMAT(1H1,/)                                                     SST00520
      WRITE(6,1110)NG,K                                                 SST00530
 1110 FORMAT(1H0,10X,'FLUXES FOR GROUP ',I2,' , PLANE 'I2)              SST00540
      IF(IOPT.EQ.1)READ(IOPO)P2                                         SST00550
      JMS=1                                                             SST00560
      JME=JM                                                            SST00570
      IF(JM.GT.50)JME=50                                                SST00580
      ITEMP2=50/JME                                                     SST00590
      ITEMP5=ITEMP2                                                     SST00591
      DO 140 I=1,IM,10                                                  SST00600
      IS=I                                                              SST00610
      IE=I+9                                                            SST00620
      IF(IE.GT.IM)IE=IM                                                 SST00630
      IF((I-1)/10.LT.ITEMP5)GO TO 110                                   SST00640
      WRITE(6,1100)                                                     SST00650
      ITEMP5=ITEMP5+50/ITEMP2                                           SST00660
  110 WRITE(6,1120)(ITEMP3,ITEMP3=IS,IE)                                SST00670
 1120 FORMAT(1H0,3X,'J / I',2X,I7,9I12)                                 SST00680
      DO 130 ITEMP3=JMS,JME                                             SST00690
      J=JME+1-ITEMP3                                                    SST00700
      IF(IOPT.EQ.1)GO TO 120                                            SST00710
      WRITE(6,1130)J,(PSI(NG,II,J,K),II=IS,IE)                          SST00720
 1130 FORMAT(1H ,2X,I2,6X,1P10D12.5)                                    SST00730
      GO TO 130                                                         SST00740
  120 WRITE(6,1130)J,(P2(II,J),II=IS,IE)                                SST00750
```

```
 130 CONTINUE
     IF(JME.GE.JM)GO TO 140
     JMS=JME+1
     JME=JMS+49
     IF(JME.GT.JM)JME=JM
     WRITE(6,1100)
     GO TO 110
 140 CONTINUE
     IF(IFLOUT.NE.2)GO TO 160
     IF(IOPT.EQ.1)GO TO 150
     WRITE(7,1140)((PSI(NG,I,J,K),I=1,IM),J=1,JM)
1140 FORMAT(5D16.10)
     GO TO 160
 150 WRITE(7,1140)((P2(I,J),I=1,IM),J=1,JM)
 160 CONTINUE
 170 CONTINUE
     IF(IOPT.EQ.1)REWIND IOPO
     IF(IFLOUT.LT.3)GO TO 220
 180 REWIND IOPSI
     DO 210 NG=1,NNG
     DO 200 K=1,KM
     IF(IOPT.EQ.1)GO TO 190
     WRITE(IOPSI)((PSI(NG,I,J,K),I=1,IM),J=1,JM)
     GO TO 200
 190 READ(IOPO)P2
     WRITE(IOPSI)P2
 200 CONTINUE
 210 CONTINUE
     IF(IOPT.EQ.1)REWIND IOPO
     REWIND IOPSI
 220 RETURN
     END
```

```
      SUBROUTINE SETUP1(V,XI,XNU,SIGF,SIGR,SIGT,SIGS,X,Y,Z,HX,HY,HZ,IBP,SET00010
     1JBP,KBP,DD1,DD2,DD3,DD4,DD5,DD6,DD7,VO,MMAP,NPRMP,NNGV,NDGV,NDNSCVSET00020
     2,NMATV,IMV,JMV,KMV,IRMV,JRMV,KRMV,NPRGV,NGXV)                     SET00030
      IMPLICIT REAL*8 (A-H,O-Z)                                         SET00040
      INTEGER*2 MMAP,NPRMP                                              SET00050
      COMMON/INTS/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   SET00100
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    SET00110
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,SET00120
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     SET00130
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    SET00140
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    SET00150
     1TEMP5,TEMP5,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           SET00160
      DIMENSION V(NNGV),XI(NNGV),XNU(NMATV,NNGV),                       SET00170
     1SIGF(NMATV,NNGV),SIGR(NMATV,NNGV),SIGT(NMATV,NNGV),SIGS(NMATV,NNGVSET00180
     2,NDNSCV),X(IMV),Y(JMV),Z(KMV),VO(NPRGV),                          SET00190
     3HX(IRMV),HY(JRMV),HZ(KRMV),IBP(IRMV),JBP(JRMV),KBP(KRMV),DD1(NPRGVSET00200
     4,NNGV),DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),DD5(NPRGV,NSET00210
     5NGV),DD6(NPRGV,NNGV),DD7(NPRGV,NGXV,NDNSCV),MMAP(IRMV,JRMV,KRMV), SET00220
     6NPRMP(IMV,JMV,KMV)                                                SET00230
      DIMENSION HD(6),MN(8)                                             SET00240
      DO 102 NM=1,NMAT                                                  SET00250
      DO 102 NG=1,NNG                                                   SET00260
      DO 101 NDN=1,NDNSCT                                               SET00270
      SIGR(NM,NG)=SIGR(NM,NG)+SIGS(NM,NG,NDN)                           SET00280
  101 CONTINUE                                                          SET00290
  102 CONTINUE                                                          SET00300
C     START WITH NESTED DO LOOPS OVER MATERIAL REGIONS                  SET00310
      ITEMP1=1                                                          SET00320
      DO 560 KR=1,KRM                                                   SET00330
      ITEMP2=1                                                          SET00340
      DO 550 JR=1,JRM                                                   SET00350
      ITEMP3=1                                                          SET00360
      DO 540 IR=1,IRM                                                   SET00370
C     HOMOGENEOUS REGION                                                SET00380
      NPR=4*IRM*JRM*(2*KR-1)+2*IRM*(2*JR-1)+2*IR                        SET00390
      KF=KBP(KR)-1                                                      SET00400
```

```
      KS=KBP(KR-1)+1
      IF(KR.EQ.1)KS=2
      JE=JBP(JR)-1
      JS=JBP(JR-1)+1
      IF(JR.EQ.1)JS=2
      IE=IBP(IR)-1
      IS=IBP(IR-1)+1
      IF(IR.EQ.1)IS=2
      ITEMP5=1
      NPRP=NPR
      GO TO 500
  110 HD(1)=HZ(KR)
      HD(2)=HD(1)
      HD(3)=HY(JR)
      HD(4)=HD(3)
      HD(5)=HX(IR)
      HD(6)=HD(5)
      DO 120 ITEMP4=1,8
      MN(ITEMP4)=MMAP(IR,JR,KR)
  120 CONTINUE
      GO TO 530
C   LOWER LEFT EDGE
  130 NPRP=NPR-4*IRM*JRM-1
      IS=IS-1
      IE=IS
      KS=KS-1
      KE=KS
      ITEMP5=2
      GO TO 500
  140 HD(1)=HZ(KR)
      HD(2)=HZ(KR-1)
      IF(KR.EQ.1)HD(2)=HD(1)
      HD(5)=HX(IR)
      HD(6)=HX(IR-1)
      IF(IR.EQ.1)HD(6)=HD(5)
      MN(1)=MMAP(IR,JR,KR-1)
```

```
      IF(KR.EQ.1)MN(1)=MN(5)
      MN(4)=MN(1)
      MN(6)=MMAP(IR-1,JR,KR)
      IF(IR.EQ.1)MN(6)=MN(5)
      MN(7)=MN(6)
      MN(2)=MMAP(IR-1,JR,KR-1)
      IF(KR.EQ.1)MN(2)=MN(6)
      IF(IR.EQ.1)MN(2)=MN(1)
      MN(3)=MN(2)
      GO TO 530
C    LEFT SIDE
  150 NPRP=NPR-1
      KE=KBP(KR)-1
      KS=KS+1
      ITEMP5=3
      GO TO 500
  160 HD(2)=HD(1)
      MN(4)=MN(8)
      MN(1)=MN(5)
      MN(2)=MN(6)
      MN(3)=MN(7)
      GO TO 530
C    LEFT FRONT EDGE
  170 NPRP=NPR-2*IRM-1
      JS=JS-1
      JE=JS
      ITEMP5=4
      GO TO 500
  180 MN(8)=MMAP(IR,JR-1,KR)
      IF(JR.EQ.1)MN(8)=MN(5)
      MN(4)=MN(8)
      MN(7)=MMAP(IR-1,JR-1,KR)
      IF(IR.EQ.1)MN(7)=MN(8)
      IF(JR.EQ.1)MN(7)=MN(6)
      MN(3)=MN(7)
      HD(4)=HY(JR-1)
```

```
      IF(JR.EQ.1)HD(4)=HD(3)                                            SET01120
      GO TO 530                                                         SET01130
C   LOWER FRONT EDGE                                                    SET01140
  190 KS=KS-1                                                           SET01150
      KE=KS                                                             SET01160
      IS=IS+1                                                           SET01170
      IE=IBP(IR)-1                                                      SET01180
      NPRP=NPR-4*IRM*JRM-2*IRM                                          SET01190
      ITEMP5=5                                                          SET01200
      GO TO 500                                                         SET01210
  200 HD(2)=HZ(KR-1)                                                    SET01220
      IF(KR.EQ.1)HD(2)=HD(1)                                            SET01230
      HD(6)=HD(5)                                                       SET01240
      MN(6)=MN(5)                                                       SET01260
      MN(1)=MMAP(IR,JR,KR-1)                                            SET01270
      IF(KR.EQ.1)MN(1)=MN(5)                                            SET01280
      MN(2)=MN(1)                                                       SET01290
      MN(7)=MN(8)                                                       SET01300
      MN(4)=MMAP(IR,JR-1,KR-1)                                          SET01310
      IF(KR.EQ.1)MN(4)=MN(8)                                            SET01320
      IF(JR.EQ.1)MN(4)=MN(1)                                            SET01330
      MN(3)=MN(4)                                                       SET01340
      GO TO 530                                                         SET01350
C   LOWER FRONT LEFT CORNER                                             SET01360
  210 NPRP=NPR-4*IRM*JRM-2*IRM-1                                        SET01370
      IS=IS-1                                                           SET01380
      IE=IS                                                             SET01390
      ITEMP5=6                                                          SET01400
      GO TO 500                                                         SET01410
  220 HD(6)=HX(IR-1)                                                    SET01420
      IF(IR.EQ.1)HD(6)=HD(5)                                            SET01430
      MN(6)=MMAP(IR-1,JR,KR)                                            SET01440
      IF(IR.EQ.1)MN(6)=MN(5)                                            SET01450
      MN(2)=MMAP(IR-1,JR,KR-1)                                          SET01460
      IF(IR.EQ.1)MN(2)=MN(1)                                            SET01470
      IF(KR.EQ.1)MN(2)=MN(6)                                            SET01480
```

```
      MN(7)=MMAP(IR-1,JR-1,KR)                                          SET01490
      IF(IR.EQ.1)MN(7)=MN(8)                                            SET01500
      IF(JR.EQ.1)MN(7)=MN(6)                                            SET01510
      MN(3)=MMAP(IR-1,JR-1,KR-1)                                        SET01520
      IF(IR.EQ.1)MN(3)=MN(4)                                            SET01530
      IF(JR.EQ.1)MN(3)=MN(2)                                            SET01540
      IF(KR.EQ.1)MN(3)=MN(7)                                            SET01550
      GO TO 530                                                         SET01560
C    FRONT SIDE                                                         SET01570
  230 NPRP=NPR-2*IRM                                                    SET01580
      IS=IS+1                                                           SET01590
      IE=IBP(IR)-1                                                      SET01600
      KS=KS+1                                                           SET01610
      KE=KBP(KR)-1                                                      SET01620
      ITEMP5=7                                                          SET01630
      GO TO 500                                                         SET01640
  240 HD(2)=HD(1)                                                       SET01650
      HD(6)=HD(5)                                                       SET01660
      MN(4)=MN(8)                                                       SET01670
      MN(7)=MN(8)                                                       SET01680
      MN(3)=MN(8)                                                       SET01690
      MN(6)=MN(5)                                                       SET01700
      MN(1)=MN(5)                                                       SET01710
      MN(2)=MN(5)                                                       SET01720
      GO TO 530                                                         SET01730
C    BOTTOM SIDE                                                        SET01740
  250 NPRP=NPR-4*IRM*JRM                                                SET01750
      KS=KS-1                                                           SET01760
      KE=KS                                                             SET01770
      JS=JS+1                                                           SET01780
      JE=JBP(JR)-1                                                      SET01790
      ITEMP5=8                                                          SET01800
      GO TO 500                                                         SET01805
  260 HD(2)=HZ(KR-1)                                                    SET01810
      IF(KR.EQ.1)HD(2)=HD(1)                                            SET01820
      HD(4)=HD(3)                                                       SET01830
```

```
      MN(8)=MN(5)
      MN(7)=MN(6)
      MN(1)=MMAP(IR,JR,KR-1)
      IF(KR.EQ.1)MN(1)=MN(5)
      MN(2)=MN(1)
      MN(3)=MN(1)
      MN(4)=MN(1)
      GO TO 530
  500 DO 520 K=KS,KE
      DO 520 J=JS,JE
      DO 510 I=IS,IE
      NPRMP(I,J,K)=NPRP
  510 CONTINUE
  520 CONTINUE
      GO TO (110,140,160,180,200,220,240,260),ITEMP5
  530 CALL COEF1(XNU,SIGF,SIGR,SIGT,SIGS,DD1,DD2,DD3,DD4,DD5,DD6,DD7,V0,
     1NNG,NDNSCT,NMAT,NPRG,HD,MN,NPRP,NGX)
      GO TO (130,150,170,190,210,230,250,540),ITEMP5
  540 CONTINUE
  550 CONTINUE
  560 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE COEF1(XNU,SIGF,SIGR,SIGT,SIGS,DD1,DD2,DD3,DD4,DD5,      COE00010
     1DD6,DD7,VO,NNGV,NDNSCV,NMATV,NPRGV,HD,MN,NPRP,NGXV)                COE00020
      IMPLICIT REAL*8 (A-H,O-Z)                                          COE00030
      INTEGER*2 MMAP,NPRMP                                               COE00040
      COMMON/INT3/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,    COE00090
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),     COE00100
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,COE00110
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,      COE00120
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                     COE00130
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,     COE00140
     1TEMP5,TEMP5,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT            COE00150
      DIMENSION XNU(NMATV,NNGV),SIGF(NMATV,NNGV),SIGR(NMATV,NNGV),SIGT(NCOE00160
     1MATV,NNGV),SIGS(NMATV,NNGV,NDNSCV),DD1(NPRGV,NNGV),                COE00170
     2DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),DD5(NPRGV,NNGV),   COE00180
     3DD6(NPRGV,NNGV),DD7(NPRGV,NGXV,NDNSCV),VO(NPRGV)                   COE00190
      DIMENSION HD(6),MN(8)                                              COE00200
C     LOOP OVER ALL GROUPS                                               COE00210
      TEMP=1.2D01                                                        COE00220
      TEMP1=8.0D0                                                        COE00230
      NPR=NPRP                                                           COE00240
      VO(NPR)=((HD(1)+HD(2))*(HD(3)+HD(4))*(HD(5)+HD(5)))/TEMP1          COE00241
      DO 110 NG=1,NNG                                                    COE00250
      DD1(NPR,NG)=((HD(3)*HD(2)/SIGT(MN(1),NG))+(HD(3)*HD(1)/SIGT(MN(5),COE00260
     1NG))+(HD(4)*HD(2)/SIGT(MN(4),NG))+(HD(4)*HD(1)/SIGT(MN(8),NG)))/(HCOE00270
     2D(5)*TEMP)                                                         COE00280
      DD2(NPR,NG)=((HD(5)*HD(2)/SIGT(MN(1),NG))+(HD(6)*HD(2)/SIGT(MN(2),COE00290
     1NG))+(HD(6)*HD(1)/SIGT(MN(6),NG))+(HD(5)*HD(1)/SIGT(MN(5),NG)))/(HCOE00300
     2D(3)*TEMP)                                                         COE00310
      DD3(NPR,NG)=((HD(4)*HD(6)/SIGT(MN(7),NG))+(HD(4)*HD(5)/SIGT(MN(8),COE00320
     1NG))+(HD(3)*HD(5)/SIGT(MN(5),NG))+(HD(3)*HD(6)/SIGT(MN(6),NG)))/(HCOE00330
     2D(1)*TEMP)                                                         COE00340
      DD4(NPR,NG)=DD1(NPR,NG)+DD2(NPR,NG)+DD3(NPR,NG)+((HD(3)*HD(2)/SIGTCOE00350
     1(MN(2),NG))+(HD(4)*HD(2)/SIGT(MN(3),NG))+(HD(4)*HD(1)/SIGT(MN(7),NCOE00360
     2G))+(HD(3)*HD(1)/SIGT(MN(6),NG)))/(HD(6)*TEMP)+((HD(6)*HD(2)/SIGT(COE00370
     3MN(3),NG))+(HD(5)*HD(2)/SIGT(MN(4),NG))+(HD(5)*HD(1)/SIGT(MN(8),NGCOE00380
     4))+(HD(5)*HD(1)/SIGT(MN(7),NG)))/(HD(4)*TEMP)+((HD(5)*HD(3)/SIGT(MCOE00390
```

```
     5N(1),NG))+(HD(3)*HD(6)/SIGT(MN(2),NG))+(HD(4)*HD(6)/SIGT(MN(3),NG)COE00400
     6)+(HD(4)*HD(5)/SIGT(MN(4),NG)))/(HD(2)*TEMP)                        COE00410
      DD5(NPR,NG)=DD4(NPR,NG)+(HD(5)*HD(3)*HD(2)*SIGR(MN(1),NG)+HD(6)*HDCOE00420
     1(3)*HD(2)*SIGR(MN(2),NG)+HD(6)*HD(4)*HD(2)*SIGR(MN(3),NG)+HD(5)*HDCOE00430
     2(4)*HD(2)*SIGR(MN(4),NG)+HD(5)*HD(3)*HD(1)*SIGR(MN(5),NG)+HD(6)*HDCOE00440
     3(3)*HD(1)*SIGR(MN(6),NG)+HD(6)*HD(4)*HD(1)*SIGR(MN(7),NG)+HD(5)*HDCOE00450
     4(4)*HD(1)*SIGR(MN(8),NG))/TEMP1                                     COE00460
      DD6(NPR,NG)=(HD(5)*HD(3)*HD(2)*SIGF(MN(1),NG)*XNU(MN(1),NG)+HD(6)*COE00470
     1HD(3)*HD(2)*SIGF(MN(2),NG)*XNU(MN(2),NG)+HD(6)*HD(4)*HD(2)*SIGF(MNCOE00480
     2(3),NG)*XNU(MN(3),NG)+HD(5)*HD(4)*HD(2)*SIGF(MN(4),NG)*XNU(MN(4),NCOE00490
     3G)+HD(5)*HD(3)*HD(1)*SIGF(MN(5),NG)*XNU(MN(5),NG)+HD(6)*HD(3)*HD(1COE00500
     4)*SIGF(MN(6),NG)*XNU(MN(6),NG)+HD(6)*HD(4)*HD(1)*SIGF(MN(7),NG)*XNCOE00510
     5U(MN(7),NG)+HD(5)*HD(4)*HD(1)*SIGF(MN(8),NG)*XNU(MN(8),NG))/TEMP1 COE00520
      IF(NG.EQ.NNG)GO TO 110                                              COE00521
      DO 100 NDN=1,NDNSCT                                                 COE00530
      DD7(NPR,NG,NDN)=(HD(5)*HD(3)*HD(2)*SIGS(MN(1),NG,NDN)+HD(6)*HD(3)*COE00540
     1HD(2)*SIGS(MN(2),NG,NDN)+HD(6)*HD(4)*HD(2)*SIGS(MN(3),NG,NDN)+HD(5COE00550
     2)*HD(4)*HD(2)*SIGS(MN(4),NG,NDN)+HD(5)*HD(3)*HD(1)*SIGS(MN(5),NG,NCOE00560
     3DN)+HD(6)*HD(3)*HD(1)*SIGS(MN(6),NG,NDN)+HD(6)*HD(4)*HD(1)*SIGS(MNCOE00570
     4(7),NG,NDN)+HD(5)*HD(4)*HD(1)*SIGS(MN(8),NG,NDN))/TEMP1             COE00580
  100 CONTINUE                                                            COE00590
  110 CONTINUE                                                            COE00600
      RETURN                                                              COE00610
      END                                                                 COE00620
```

```
      SUBROUTINE INNERO(X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5,MMAP,NPRMP,
     1PSI,P1,P2,P3,FO,SRC,WA,GA,SOLN,OMEG,XFISS,XINSC,XREM,XLEK,
     2NNGV,NMATV,IMV,JMV,KMV,IRMV,JRMV,KRMV,NPRGV,NG)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 MMAP,NPRMP
      COMMON/INTS/IASIZE,NNG,NDG,NTOG,NMAT,IM, JM,KM,IRM,JRM,KRM,NLBC,
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT
      DIMENSION X(IMV),Y(JMV),Z(KMV),HX(IRMV),HY(JRMV),HZ(KRMV),
     1DD1(NPRGV,NNGV),DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),
     2DD5(NPRGV,NNGV),MMAP(IRMV,JRMV,KRMV),NPRMP(IMV,JMV,KMV),
     3PSI(NNGV,IMV,JMV,KMV),P1(IMV,JMV),P2(IMV,JMV),P3(IMV,JMV),
     4SRC(IMV,JMV,KMV),WA(IMV),GA(IMV),SOLN(IMV),OMEG(NNGV),XFISS(NNGV),
     5XINSC(NNGV),XREM(NNGV),XLEK(NNGV)
      IF(ITEMP5.EQ.5)GO TO 90
      CALL GRBALO(DD1,DD2,DD3,DD4,DD5,NPRMP,PSI,P1,P2,P3,XFISS,XINSC,
     1XREM,XLEK,NNG,IM,JM,KM,NPRG,NG)
   90 NIT=0
C    START WITH BOTTOM PLANE
  100 CONTINUE
      XNLBC=NLBC
      TEMP1=0.0D0
      TEMP4=1.0D+50
      K=1
      IF(NBBC.EQ.0)GO TO 200
      J=1
      IF(NFBC.EQ.0)GO TO 140
      NPR=NPRMP(1,1,1)
      WA(1)=-2.0D0*DD1(NPR,NG)*XNLBC/DD5(NPR,NG)
      GA(1)=((SRC(1,J,K)+2.0D0*(DD2(NPR,NG)*PSI(NG,1,2,1)+DD3(NPR,NG)*PS
     1I(NG,1,1,2)))*XNLBC)/DD5(NPR,NG)
      DO 110 I=2,IMX
```

```
      NPR=NPRMP(I,1,1)
      NPRX=NPRMP(I-1,1,1)
      TEMP=1.0D0/(DD5(NPR,NG)+DD1(NPRX,NG)*WA(I-1))
      WA(I)=-DD1(NPR,NG)*TEMP
  110 GA(I)=(SRC(I,J,K)+2.0D0*(DD2(NPR,NG)*PSI(NG,I,2,1)+DD3(NPR,NG)*PSI
     1(NG,I,1,2))+DD1(NPRX,NG)*GA(I-1))*TEMP
      SOLN(IM)=0.0D0
      DO 120 II=1,IMX
      I=IM-II
  120 SOLN(I)=GA(I)-WA(I)*SOLN(I+1)
      DO 130 I=1,IM
      TEMP2=PSI(NG,I,1,1)
      PSI(NG,I,1,1)=TEMP2+OMEG(NG)*(SOLN(I)-TEMP2)
      IF(PSI(NG,I,1,1).GT.0.0D0)GO TO 125
      PSI(NG,I,1,1)=0.0D0
      GO TO 130
  125 TEMP3=TEMP2/PSI(NG,I,1,1)
      IF(TEMP1.LT.TEMP3)TEMP1=TEMP3
      IF(TEMP4.GT.TEMP3)TEMP4=TEMP3
  130 CONTINUE
  140 DO 180 J=2,JMX
      NPR=NPRMP(1,J,1)
      NPRY=NPRMP(1,J-1,1)
      WA(1)=-2.0D0*DD1(NPR,NG)*XNLBC/DD5(NPR,NG)
      GA(1)=((SRC(1,J,K)+DD2(NPRY,NG)*PSI(NG,1,J-1,1)+DD2(NPR,NG)*PSI(NG
     1,1,J+1,1)+2.0D0*DD3(NPR,NG)*PSI(NG,1,J,2))*XNLBC)/DD5(NPR,NG)
      DO 150 I=2,IMX
      NPR=NPRMP(I,J,1)
      NPRX=NPRMP(I-1,J,1)
      NPRY=NPRMP(I,J-1,1)
      TEMP=1.0D0/(DD5(NPR,NG)+DD1(NPRX,NG)*WA(I-1))
      WA(I)=-DD1(NPR,NG)*TEMP
  150 GA(I)=(SRC(I,J,1)+DD2(NPRY,NG)*PSI(NG,I,J-1,1)+DD2(NPR,NG)*PSI(NG,
     1I,J+1,1)+2.0D0*DD3(NPR,NG)*PSI(NG,I,J,2)+DD1(NPRX,NG)*GA(I-1))*TEM
     2P
      SOLN(IM)=0.0D0
```

```
      DO 160 II=1,IMX
      I=IM-II
  160 SOLN(I)=GA(I)-WA(I)*SOLN(I+1)
      DO 170 I=1,IM
      TEMP2=PSI(NG,I,J,1)
      PSI(NG,I,J,1)=TEMP2+OMEG(NG)*(SOLN(I)-TEMP2)
      IF(PSI(NG,I,J,1).GT.0.0D0)GO TO 165
      PSI(NG,I,J,1)=0.0D0
      GO TO 170
  165 TEMP3=TEMP2/PSI(NG,I,J,1)
      IF(TEMP1.LT.TEMP3)TEMP1=TEMP3
      IF(TEMP4.GT.TEMP3)TEMP4=TEMP3
  170 CONTINUE
  180 CONTINUE
      DO 190 I=1,IM
  190 PSI(NG,I,JM,1)=0.0D0
C   NOW COMPUTE FOR THE REST OF THE PLANES
  200 DO 300 K=2,KMX
      IF(NFBC.EQ.0) GO TO 240
      J=1
      NPR=NPRMP(1,1,K)
      NPRZ=NPRMP(1,1,K-1)
      WA(1)=-2.0D0*DD1(NPR,NG)*XNLBC/DD5(NPR,NG)
      GA(1)=((SRC(1,1,K)+2.0D0*DD2(NPR,NG)*PSI(NG,1,2,K)+DD3(NPR,NG)*PSI
     1(NG,1,1,K+1)+DD3(NPRZ,NG)*PSI(NG,1,1,K-1))*XNLBC)/DD5(NPR,NG)
      DO 210 I=2,IMX
      NPR=NPRMP(I,1,K)
      NPRX=NPRMP(I-1,1,K)
      NPRZ=NPRMP(I,1,K-1)
      TEMP=1.0D0/(DD5(NPR,NG)+DD1(NPRX,NG)*WA(I-1))
      WA(I)=-DD1(NPR,NG)*TEMP
  210 GA(I)=(SRC(I,1,K)+2.0D0*DD2(NPR,NG)*PSI(NG,I,2,K)+DD3(NPR,NG)*PSI(
     1NG,I,1,K+1)+DD3(NPRZ,NG)*PSI(NG,I,1,K-1)+DD1(NPRX,NG)*GA(I-1))*TEM
     2P
      SOLN(IM)=0.0D0
      DO 220 II=1,IMX
```

```
      I=IM-II
  220 SOLN(I)=GA(I)-WA(I)*SOLN(I+1)
      DO 230 I=1,IM
      TEMP2=PSI(NG,I,1,K)
      PSI(NG,I,1,K)=TEMP2+OMEG(NG)*(SOLN(I)-TEMP2)
      IF(PSI(NG,I,1,K).GT.0.0D0)GO TO 225
      PSI(NG,I,1,K)=0.0D0
      GO TO 230
  225 TEMP3=TEMP2/PSI(NG,I,1,K)
      IF(TEMP1.LT.TEMP3)TEMP1=TEMP3
      IF(TEMP4.GT.TEMP3)TEMP4=TEMP3
  230 CONTINUE
  240 DO 280 J=2,JMX
      NPR=NPRMP(1,J,K)
      NPRY=NPRMP(1,J-1,K)
      NPRZ=NPRMP(1,J,K-1)
      WA(1)=-2.0D0*DD1(NPR,NG)*XNLBC/DD5(NPR,NG)
      GA(1)=((SRC(1,J,K)+DD2(NPR,NG)*PSI(NG,1,J+1,K)+DD2(NPRY,NG)*PSI(NG
     1,1,J-1,K)+DD3(NPR,NG)*PSI(NG,1,J,K+1)+DD3(NPRZ,NG)*PSI(NG,1,J,K-1)
     2)*XNLBC)/DD5(NPR,NG)
      DO 250 I=2,IMX
      NPR=NPRMP(I,J,K)
      NPRX=NPRMP(I-1,J,K)
      NPRY=NPRMP(I,J-1,K)
      NPRZ=NPRMP(I,J,K-1)
      TEMP=1.0D0/(DD5(NPR,NG)+DD1(NPRX,NG)*WA(I-1))
      WA(I)=-DD1(NPR,NG)*TEMP
  250 GA(I)=(SRC(I,J,K)+DD2(NPR,NG)*PSI(NG,I,J+1,K)+DD2(NPRY,NG)*PSI(NG,
     1I,J-1,K)+DD3(NPR,NG)*PSI(NG,I,J,K+1)+DD3(NPRZ,NG)*PSI(NG,I,J,K-1)+
     2DD1(NPRX,NG)*GA(I-1))*TEMP
      SOLN(IM)=0.0D0
      DO 260 II=1,IMX
      I=IM-II
  260 SOLN(I)=GA(I)-WA(I)*SOLN(I+1)
      DO 270 I=1,IM
      TEMP2=PSI(NG,I,J,K)
```

```
      PSI(NG,I,J,K)=TEMP2+OMEG(NG)*(SOLN(I)-TEMP2)
      IF(PSI(NG,I,J,K).GT.0.0D0)GO TO 265
      PSI(NG,I,J,K)=0.0D0
      GO TO 270
  265 TEMP3=TEMP2/PSI(NG,I,J,K)
      IF(TEMP1.LT.TEMP3)TEMP1=TEMP3
      IF(TEMP4.GT.TEMP3)TEMP4=TEMP3
  270 CONTINUE
  280 CONTINUE
      DO 290 I=1,IM
  290 PSI(NG,I,JM,K)=0.0D0
  300 CONTINUE
C   COMPLETE MESH NOW SWEPT
C   NOW COMPUTE LARGEST RESIDUAL
      TEMP2=DABS(1.0D0-TEMP1)
      TEMP3=DABS(1.0D0-TEMP4)
      IF(TEMP2-TEMP3)320,320,310
  310 TEMP3=TEMP2
  320 NTIT=NTIT+1
      NIT=NIT+1
      IF(NIT.GE.NIIT)GO TO 330
      IF(TEMP3.GT.EPS2)GO TO 100
  330 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GRBALO(DD1,DD2,DD3,DD4,DD5,NPRMP,PSI,P1,P2,P3,XFISS,
     1XINSC,XREM,XLEK,NNGV,IMV,JMV,KMV,NPRGV,NG)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 MMAP,NPRMP
      COMMON/INTS/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT
      DIMENSION DD1(NPRGV,NNGV),DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRG
     1V,NNGV),DD5(NPRGV,NNGV),NPRMP(IMV,JMV,KMV),PSI(NNGV,IMV,JMV,KMV),
     2P1(IMV,JMV),P2(IMV,JMV),P3(IMV,JMV),XFISS(NNGV),XINSC(NNGV),
     3XREM(NNGV),XLEK(NNGV)
      XREM(NG)=0.0D0
      XLEK(NG)=0.0D0
      ONE=1.0D0
      HALF=0.5D0
      VOLB=ONE
      VOLF=ONE
      VOLL=ONE
      IF(NBBC.EQ.1)VOLB=HALF
      DO 230 K=1,KMX
      IF(K.GT.1)VOLB=ONE
      IF(K.EQ.1.AND.NBBC.EQ.0) GO TO 230
      IF(K.NE.2)GO TO 120
  100 IF(NBBC.EQ.1)GO TO 120
C   COMPUTE LEAKAGE FOR BOTTOM PLANE
      IF(NFBC.EQ.1)VOLF=HALF
      DO 110 J=1,JMX
      IF(J.GT.1)VOLF=ONE
      IF(NLBC.EQ.1)VOLL=HALF
      DO 110 I=1,IMX
      IF(I.GT.1)VOLL=ONE
      NPR=NPRMP(I,J,1)
```

```
      XLEK(NG)=XLEK(NG)+DD3(NPR,NG)*PSI(NG,I,J,2)*VOLF*VOLL                 GRB00300
  110 CONTINUE                                                              GRB00310
C     COMPUTE FRONT LEAKAGE                                                 GRB00320
  120 IF(NFBC.EQ.1) GO TO 140                                               GRB00330
      IF(NLBC.EQ.1)VOLL=HALF                                                GRB00331
      DO 130 I=1,IMX                                                        GRB00340
      IF(I.GT.1)VOLL=ONE                                                    GRB00341
      NPR=NPRMP(I,1,K)                                                      GRB00350
  130 XLEK(NG)=XLEK(NG)+DD2(NPR,NG)*PSI(NG,I,2,K)*VOLL*VOLB                 GRB00360
C     COMPUTE LEFT LEAKAGE                                                  GRB00370
  140 IF(NLBC.EQ.1) GO TO 160                                               GRB00380
      IF(NFBC.EQ.1)VOLF=HALF                                                GRB00381
      DO 150 J=1,JMX                                                        GRB00390
      IF(J.GT.1)VOLF=ONE                                                    GRB00391
      NPR=NPRMP(1,J,K)                                                      GRB00400
  150 XLEK(NG)=XLEK(NG)+DD1(NPR,NG)*PSI(NG,2,J,K)*VOLF*VOLB                 GRB00410
C     COMPUTE RIGHT LEAKAGE                                                 GRB00420
  160 IF(NFBC.EQ.1)VOLF=HALF                                                GRB00421
      DO 170 J=1,JMX                                                        GRB00430
      IF(J.GT.1)VOLF=ONE                                                    GRB00431
      NPR=NPRMP(IMX,J,K)                                                    GRB00440
  170 XLEK(NG)=XLEK(NG)+DD1(NPR,NG)*PSI(NG,IMX,J,K)*VOLF*VOLB               GRB00450
C     COMPUTE BACK LEAKAGE                                                  GRB00460
      IF(NLBC.EQ.1)VOLL=HALF                                                GRB00461
      DO 180 I=1,IMX                                                        GRB00470
      IF(I.GT.1)VOLL=ONE                                                    GRB00471
      NPR=NPRMP(I,JMX,K)                                                    GRB00480
  180 XLEK(NG)=XLEK(NG)+DD2(NPR,NG)*PSI(NG,I,JMX,K)*VOLL*VOLB               GRB00490
      IF(NFBC.EQ.1)VOLF=HALF                                                GRB00491
      DO 200 J=1,JMX                                                        GRB00500
      IF(J.GT.1)VOLF=ONE                                                    GRB00501
      VOLC=VOLB*VOLF                                                        GRB00502
      IF(NLBC.EQ.1)VOLL=HALF                                                GRB00503
      DO 190 I=1,IMX                                                        GRB00510
      IF(I.GT.1)VOLL=ONE                                                    GRB00511
      VOLD=VOLL*VOLC                                                        GRB00512
```

```
      NPR=NPRMP(I,J,K)                                                  GRB00520
  190 XREM(NG)=XREM(NG)+(DD5(NPR,NG)-DD4(NPR,NG))*PSI(NG,I,J,K)*VOLD    GRB00530
  200 CONTINUE                                                          GRB00540
      IF(K.LT.KMX) GO TO 230                                            GRB00550
C   COMPUTE TOP LEAKAGE                                                 GRB00560
      IF(NFBC.EQ.1)VOLF=HALF                                            GRB00561
      DO 220 J=1,JMX                                                    GRB00570
      IF(J.GT.1)VOLF=ONE                                                GRB00571
      IF(NLBC.EQ.1)VOLL=HALF                                            GRB00572
      DO 210 I=1,IMX                                                    GRB00580
      IF(I.GT.1)VOLL=ONE                                                GRB00581
      NPR=NPRMP(I,J,KMX)                                                GRB00590
  210 XLEK(NG)=XLEK(NG)+DD3(NPR,NG)*PSI(NG,I,J,KMX)*VOLL*VOLF           GRB00600
  220 CONTINUE                                                          GRB00610
  230 CONTINUE                                                          GRB00620
      TEMP=(XFISS(NG)+XINSC(NG))/(XLEK(NG)+XREM(NG))                    GRB00630
      DO 250 K=1,KMX                                                    GRB00640
      DO 250 J=1,JM                                                     GRB00650
      DO 240 I=1,IM                                                     GRB00660
  240 PSI(NG,I,J,K)=TEMP*PSI(NG,I,J,K)                                  GRB00670
  250 CONTINUE                                                          GRB00680
      XREM(NG)=TEMP*XREM(NG)                                            GRB00690
      XLEK(NG)=TEMP*XLEK(NG)                                            GRB00700
      RETURN                                                            GRB00710
      END                                                               GRB00720
```

```
      SUBROUTINE INNER1(X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5,MMAP,NPRMP,  INN00010
     1PSI,P1,P2,P3,FO,SRC,WA,GA,SOLN,OMEG,XFISS,XINSC,XREM,XLEK,        INN00020
     2NNGV,NMATV,IMV,JMV,KMV,IRMV,JRMV,KRMV,NPRGV,NG)                   INN00030
      IMPLICIT REAL*8 (A-H,O-Z)                                          INN00040
      INTEGER*2 MMAP,NPRMP                                               INN00050
      COMMON/INTS/IASIZE,NNG,NDG,NTOG,NMAT,IM, JM,KM,IRM,JRM,KRM,NLBC,   INN00100
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),     INN00110
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,INN00120
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,      INN00130
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                     INN00140
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,     INN00150
     1TEMP5,TEMP5,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT            INN00160
      DIMENSION X(IMV),Y(JMV),Z(KMV),HX(IRMV),HY(JRMV),HZ(KRMV),         INN00170
     1DD1(NPRGV,NNGV),DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),   INN00180
     2DD5(NPRGV,NNGV),MMAP(IRMV,JRMV,KRMV),NPRMP(IMV,JMV,KMV),           INN00190
     3PSI(NNGV,IMV,JMV,KMV),P1(IMV,JMV),P2(IMV,JMV),P3(IMV,JMV),         INN00200
     4SRC(IMV,JMV,KMV),WA(IMV),GA(IMV),SOLN(IMV),OMEG(NNGV),XFISS(NNGV),INN00210
     5XINSC(NNGV),XREM(NNGV),XLEK(NNGV)                                  INN00220
      IF(ITEMP5.EQ.5)GO TO 90                                            INN00230
      CALL GRBAL1(DD1,DD2,DD3,DD4,DD5,NPRMP,PSI,P1,P2,P3,XFISS,XINSC,    INN00240
     1XREM,XLEK,NNG,IM,JM,KM,NPRG,NG)                                    INN00250
   90 NIT=0                                                              INN00260
      XNLBC=NLBC                                                         INN00270
C   START WITH BOTTOM PLANE                                              INN00280
  100 CONTINUE                                                           INN00290
      REWIND IOSC1                                                       INN00300
      REWIND IOSC2                                                       INN00310
      TEMP1=0.0D0                                                        INN00320
      TEMP4=1.0D+50                                                      INN00330
      K=1                                                                INN00340
      READ(IOSC1)P1                                                      INN00350
      READ(IOSC1)P2                                                      INN00360
      IF(NBBC.EQ.0)GO TO 200                                             INN00370
      DO 185 NP=1,NPIT                                                   INN00380
      IF(NP.LT.NPIT)GO TO 105                                            INN00390
      TEMP1=0.0D0                                                        INN00400
```

```
      TEMP4=1.0D+50
  105 J=1
      IF(NFBC.EQ.0)GO TO 140
      NPR=NPRMP(1,1,1)
      WA(1)=-2.0D0*DD1(NPR,NG)*XNLBC/DD5(NPR,NG)
      GA(1)=((SRC(1,J,K)+2.0D0*(DD2(NPR,NG)*P1(1,2)+DD3(NPR,NG)*P2(1,1))
     1)*XNLBC)/DD5(NPR,NG)
      DO 110 I=2,IMX
      NPR=NPRMP(I,1,1)
      NPRX=NPRMP(I-1,1,1)
      TEMP=1.0D0/(DD5(NPR,NG)+DD1(NPRX,NG)*WA(I-1))
      WA(I)=-DD1(NPR,NG)*TEMP
  110 GA(I)=(SRC(I,1,1)+2.0D0*(DD2(NPR,NG)*P1(I,2)+DD3(NPR,NG)*P2(I,1))
     1+DD1(NPRX,NG)*GA(I-1))*TEMP
      SOLN(IM)=0.0D0
      DO 120 II=1,IMX
      I=IM-II
  120 SOLN(I)=GA(I)-WA(I)*SOLN(I+1)
      DO 130 I=1,IM
      TEMP2=P1(I,1)
      P1(I,1)=TEMP2+OMEG(NG)*(SOLN(I)-TEMP2)
      IF(P1(I,1).GT.0.0D0)GO TO 125
      P1(I,1)=0.0D0
      GO TO 130
  125 TEMP3=TEMP2/P1(I,1)
      IF(TEMP1.LT.TEMP3)TEMP1=TEMP3
      IF(TEMP4.GT.TEMP3)TEMP4=TEMP3
  130 CONTINUE
  140 DO 180 J=2,JMX
      NPR=NPRMP(1,J,1)
      NPRY=NPRMP(1,J-1,1)
      WA(1)=-2.0D0*DD1(NPR,NG)*XNLBC/DD5(NPR,NG)
      GA(1)=((SRC(1,J,1)+DD2(NPRY,NG)*P1(1,J-1)+DD2(NPR,NG)*P1(1,J+1)+
     12.0D0*DD3(NPR,NG)*P2(1,J))*XNLBC)/DD5(NPR,NG)
      DO 150 I=2,IMX
      NPR=NPRMP(I,J,1)
```

```
      NPRX=NPRMP(I-1,J,1)
      NPRY=NPRMP(I,J-1,1)
      TEMP=1.0D0/(DD5(NPR,NG)+DD1(NPRX,NG)*WA(I-1))
      WA(I)=-DD1(NPR,NG)*TEMP
  150 GA(I)=(SRC(I,J,1)+DD2(NPRY,NG)*P1(I,J-1)+DD2(NPR,NG)*P1(I,J+1)+
     12.0D0*DD3(NPR,NG)*P2(I,J)+DD1(NPRX,NG)*GA(I-1))*TEMP
      SOLN(IM)=0.0D0
      DO 160 II=1,IMX
      I=IM-II
  160 SOLN(I)=GA(I)-WA(I)*SOLN(I+1)
      DO 170 I=1,IM
      TEMP2=P1(I,J)
      P1(I,J)=TEMP2+OMEG(NG)*(SOLN(I)-TEMP2)
      IF(P1(I,J).GT.0.0D0)GO TO 165
      P1(I,J)=0.0D0
      GO TO 170
  165 TEMP3=TEMP2/P1(I,J)
      IF(TEMP1.LT.TEMP3)TEMP1=TEMP3
      IF(TEMP4.GT.TEMP3)TEMP4=TEMP3
  170 CONTINUE
  180 CONTINUE
  185 CONTINUE
      TEMP5=TEMP1
      TEMP6=TEMP4
  190 P1(I,JM)=0.0D0
C   NOW COMPUTE FOR THE REST OF THE PLANES
  200 DO 310 K=2,KMX
      READ(IOSC1)P3
      DO 295 NP=1,NPIT
      IF(NP.LT.NPIT)GO TO 205
      TEMP1=0.0D0
      TEMP4=1.0D+50
  205 J=1
      IF(NFBC.EQ.0) GO TO 240
      NPR=NPRMP(1,1,K)
      NPRZ=NPRMP(1,1,K-1)
```

```
      WA(1)=-2.0D0*DD1(NPR,NG)*XNLBC/DD5(NPR,NG)
      GA(1)=((SRC(1,1,K)+2.0D0*DD2(NPR,NG)*P2(1,2)+DD3(NPR,NG)*P3(1,1)+
     1DD3(NPRZ,NG)*P1(1,1))*XNLBC)/DD5(NPR,NG)
      DO 210 I=2,IMX
      NPR=NPRMP(I,1,K)
      NPRX=NPRMP(I-1,1,K)
      NPRZ=NPRMP(I,1,K-1)
      TEMP=1.0D0/(DD5(NPR,NG)+DD1(NPRX,NG)*WA(I-1))
      WA(I)=-DD1(NPR,NG)*TEMP
  210 GA(I)=(SRC(I,1,K)+2.0D0*DD2(NPR,NG)*P2(I,2)+DD3(NPR,NG)*P3(I,1)+
     1DD3(NPRZ,NG)*P1(I,1)+DD1(NPRX,NG)*GA(I-1))*TEMP
      SOLN(IM)=0.0D0
      DO 220 II=1,IMX
      I=IM-II
  220 SOLN(I)=GA(I)-WA(I)*SOLN(I+1)
      DO 230 I=1,IM
      TEMP2=P2(I,1)
      P2(I,1)=TEMP2+OMEG(NG)*(SOLN(I)-TEMP2)
      IF(P2(I,1).GT.0.0D0)GO TO 225
      P2(I,1)=0.0D0
      GO TO 230
  225 TEMP3=TEMP2/P2(I,1)
      IF(TEMP1.LT.TEMP3)TEMP1=TEMP3
      IF(TEMP4.GT.TEMP3)TEMP4=TEMP3
  230 CONTINUE
  240 DO 280 J=2,JMX
      NPR=NPRMP(1,J,K)
      NPRY=NPRMP(1,J-1,K)
      NPRZ=NPRMP(1,J,K-1)
      WA(1)=-2.0D0*DD1(NPR,NG)*XNLBC/DD5(NPR,NG)
      GA(1)=((SRC(1,J,K)+DD2(NPR,NG)*P2(1,J+1)+DD2(NPRY,NG)*P2(1,J-1)+
     1DD3(NPR,NG)*P3(1,J)+DD3(NPRZ,NG)*P1(1,J))*XNLBC)/DD5(NPR,NG)
      DO 250 I=2,IMX
      NPR=NPRMP(I,J,K)
      NPRX=NPRMP(I-1,J,K)
      NPRY=NPRMP(I,J-1,K)
```

```
      NPRZ=NPRMP(I,J,K-1)
      TEMP=1.0D0/(DD5(NPR,NG)+DD1(NPRX,NG)*WA(I-1))
      WA(I)=-DD1(NPR,NG)*TEMP
  250 GA(I)=(SRC(I,J,K)+DD2(NPR,NG)*P2(I,J+1)+DD2(NPRY,NG)*P2(I,J-1)+
     1DD3(NPR,NG)*P3(I,J)+DD3(NPRZ,NG)*P1(I,J)+DD1(NPRX,NG)*GA(I-1))*TEM
     2P
      SOLN(IM)=0.0D0
      DO 260 II=1,IMX
      I=IM-II
  260 SOLN(I)=GA(I)-WA(I)*SOLN(I+1)
      DO 270 I=1,IM
      TEMP2=P2(I,J)
      P2(I,J)=TEMP2+OMEG(NG)*(SOLN(I)-TEMP2)
      IF(P2(I,J).GT.0.0D0)GO TO 265
      P2(I,J)=0.0D0
      GO TO 270
  265 TEMP3=TEMP2/P2(I,J)
      IF(TEMP1.LT.TEMP3)TEMP1=TEMP3
      IF(TEMP4.GT.TEMP3)TEMP4=TEMP3
  270 CONTINUE
  280 CONTINUE
      DO 290 I=1,IM
  290 P2(I,JM)=0.0D0
  295 CONTINUE
C     TEST MIN AND MAX FLUX RATIO FOR THIS PLANE
      IF(TEMP1.GT.TEMP5)TEMP5=TEMP1
      IF(TEMP4.LT.TEMP6)TEMP6=TEMP4
      WRITE(IOSC2)P1
      DO 305 J=1,JM
      DO 305 I=1,IM
      P1(I,J)=P2(I,J)
  305 P2(I,J)=P3(I,J)
  310 CONTINUE
C     COMPLETE MESH NOW SWEPT
      WRITE(IOSC2)P1
      WRITE(IOSC2)P2
```

```
C    SWITCH DATASET DESIGNATIONS
      ITEMP4=IOSC2
      IOSC2=IOSC1
      IOSC1=ITEMP4
C    NOW COMPUTE LARGEST RESIDUAL
      TEMP1=TEMP5
      TEMP4=TEMP6
      TEMP2=DABS(1.0D0-TEMP1)
      TEMP3=DABS(1.0D0-TEMP4)
      IF(TEMP2-TEMP3)330,330,320
  320 TEMP3=TEMP2
  330 NTIT=NTIT+1
      NIT=NIT+1
      IF(NIT.GE.NIIT)GO TO 340
      IF(TEMP3.GT.EPS2)GO TO 100
C    INNER ITERATION CONVERGES, WRITE FLUXES ON IOPN
  340 CONTINUE
      REWIND IOSC1
      IF(ITEMP5.EQ.5)GO TO 360
      DO 350 K=1,KM
      READ(IOSC1)P2
      WRITE(IOPN)P2
  350 CONTINUE
  360 RETURN
      END
```

```
      SUBROUTINE GRBAL1(DD1,DD2,DD3,DD4,DD5,NPRMP,PSI,P1,P2,P3,XFISS,   GRB00010
     1XINSC,XREM,XLEK,NNGV,IMV,JMV,KMV,NPRGV,NG)                        GRB00020
      IMPLICIT REAL*8 (A-H,O-Z)                                         GRB00030
      INTEGER*2 MMAP,NPRMP                                              GRB00040
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   GRB00090
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    GRB00100
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,GRB00110
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     GRB00120
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    GRB00130
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    GRB00140
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           GRB00150
      DIMENSION DD1(NPRGV,NNGV),DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGGRB00160
     1V,NNGV),DD5(NPRGV,NNGV),NPRMP(IMV,JMV,KMV),PSI(NNGV,IMV,JMV,KMV), GRB00170
     2P1(IMV,JMV),P2(IMV,JMV),P3(IMV,JMV),XFISS(NNGV),XINSC(NNGV),      GRB00180
     3XREM(NNGV),XLEK(NNGV)                                             GRB00190
      XREM(NG)=0.0D0                                                    GRB00200
      XLEK(NG)=0.0D0                                                    GRB00210
      REWIND IOSC1                                                      GRB00220
      REWIND IOSC2                                                      GRB00230
      ONE=1.0D0                                                         GRB00231
      HALF=0.5D0                                                        GRB00232
      VOLB=ONE                                                          GRB00233
      VOLF=ONE                                                          GRB00234
      VOLL=ONE                                                          GRB00235
      IF(NBBC.EQ.1)VOLB=HALF                                            GRB00236
      DO 230 K=1,KMX                                                    GRB00240
      READ(IOPO)P2                                                      GRB00250
      WRITE(IOSC2)P2                                                    GRB00260
      IF(K.GT.1)VOLB=ONE                                                GRB00261
      IF(K.EQ.1.AND.NBBC.EQ.0) GO TO 230                                GRB00270
      IF(K.NE.2)GO TO 120                                               GRB00280
  100 IF(NBBC.EQ.1)GO TO 120                                            GRB00290
C   COMPUTE LEAKAGE FOR BOTTOM PLANE                                    GRB00300
      IF(NFBC.EQ.1)VOLF=HALF                                            GRB00301
      DO 110 J=1,JMX                                                    GRB00310
      IF(J.GT.1)VOLF=ONE                                                GRB00311
```

```
      IF(NLBC.EQ.1)VOLL=HALF                                            GRB00312
      DO 110 I=1,IMX                                                    GRB00320
      IF(I.GT.1)VOLL=ONE                                                GRB00321
      NPR=NPRMP(I,J,1)                                                  GRB00330
      XLEK(NG)=XLEK(NG)+DD3(NPR,NG)*P2(I,J)*VOLF*VOLL                   GRB00340
  110 CONTINUE                                                          GRB00350
C   COMPUTE FRONT LEAKAGE                                               GRB00360
  120 IF(NFBC.EQ.1) GO TO 140                                           GRB00370
      IF(NLBC.EQ.1)VOLL=HALF                                            GRB00371
      DO 130 I=1,IMX                                                    GRB00380
      IF(I.GT.1)VOLL=ONE                                                GRB00381
      NPR=NPRMP(I,1,K)                                                  GRB00390
  130 XLEK(NG)=XLEK(NG)+DD2(NPR,NG)*P2(I,2)*VOLL*VOLB                   GRB00400
C   COMPUTE LEFT LEAKAGE                                                GRB00410
  140 IF(NLBC.EQ.1) GO TO 160                                           GRB00420
      IF(NFBC.EQ.1)VOLF=HALF                                            GRB00421
      DO 150 J=1,JMX                                                    GRB00430
      IF(J.GT.1)VOLF=ONE                                                GRB00431
      NPR=NPRMP(1,J,K)                                                  GRB00440
  150 XLEK(NG)=XLEK(NG)+DD1(NPR,NG)*P2(2,J)*VOLF*VOLB                   GRB00450
C   COMPUTE RIGHT LEAKAGE                                               GRB00460
  160 IF(NFBC.EQ.1)VOLF=HALF                                            GRB00461
      DO 170 J=1,JMX                                                    GRB00470
      IF(J.GT.1)VOLF=ONE                                                GRB00471
      NPR=NPRMP(IMX,J,K)                                                GRB00480
  170 XLEK(NG)=XLEK(NG)+DD1(NPR,NG)*P2(IMX,J)*VOLF*VOLB                 GRB00490
C   COMPUTE BACK LEAKAGE                                                GRB00500
      IF(NLBC.EQ.1)VOLL=HALF                                            GRB00501
      DO 180 I=1,IMX                                                    GRB00510
      IF(I.GT.1)VOLL=ONE                                                GRB00511
      NPR=NPRMP(I,JMX,K)                                                GRB00520
  180 XLEK(NG)=XLEK(NG)+DD2(NPR,NG)*P2(I,JMX)*VOLL*VOLB                 GRB00530
      IF(NFBC.EQ.1)VOLF=HALF                                            GRB00531
      DO 200 J=1,JMX                                                    GRB00540
      IF(J.GT.1)VOLF=ONE                                                GRB00541
      VOLC=VOLB*VOLF                                                    GRB00542
```

```
      IF(NLBC.EQ.1)VOLL=HALF                                            GRB00543
      DO 190 I=1,IMX                                                    GRB00550
      IF(I.GT.1)VOLL=ONE                                                GRB00551
      VOLD=VOLL*VOLC                                                    GRB00552
      NPR=NPRMP(I,J,K)                                                  GRB00560
  190 XREM(NG)=XREM(NG)+(DD5(NPR,NG)-DD4(NPR,NG))*P2(I,J)*VOLD          GRB00570
  200 CONTINUE                                                          GRB00580
      IF(K.LT.KMX) GO TO 230                                            GRB00590
C   COMPUTE TOP LEAKAGE                                                 GRB00600
      IF(NFBC.EQ.1)VOLF=HALF                                            GRB00601
      DO 220 J=1,JMX                                                    GRB00610
      IF(J.GT.1)VOLF=ONE                                                GRB00610
      IF(NLBC.EQ.1)VOLL=HALF                                            GRB00612
      DO 210 I=1,IMX                                                    GRB00620
      IF(I.GT.1)VOLL=ONE                                                GRB00621
      NPR=NPRMP(I,J,KMX)                                                GRB00630
  210 XLEK(NG)=XLEK(NG)+DD3(NPR,NG)*P2(I,J)*VOLL*VOLF                   GRB00640
  220 CONTINUE                                                          GRB00650
  230 CONTINUE                                                          GRB00660
      READ(IOPO)P2                                                      GRB00670
      WRITE(IOSC2)P2                                                    GRB00680
      REWIND IOSC2                                                      GRB00690
      TEMP=(XFISS(NG)+XINSC(NG))/(XLEK(NG)+XREM(NG))                    GRB00700
      DO 260 K=1,KM                                                     GRB00710
      READ(IOSC2)P2                                                     GRB00720
      DO 250 J=1,JM                                                     GRB00730
      DO 250 I=1,IM                                                     GRB00740
  250 P2(I,J)=TEMP*P2(I,J)                                              GRB00750
      WRITE(IOSC1)P2                                                    GRB00760
  260 CONTINUE                                                          GRB00770
      XREM(NG)=TEMP*XREM(NG)                                            GRB00780
      XLEK(NG)=TEMP*XLEK(NG)                                            GRB00790
      RETURN                                                            GRB00800
      END                                                               GRB00810
```

```
      SUBROUTINE FLUXTR(PSI,P2,NNGV,IMV,JMV,KMV)                        FLU00010
      IMPLICIT REAL*8 (A-H,O-Z)                                         FLU00020
      INTEGER*2 MMAP,NPRMP                                              FLU00030
      COMMON/INTS/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   FLU00040
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    FLU00050
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,FLU00060
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     FLU00070
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    FLU00080
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    FLU00130
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           FLU00140
      DIMENSION PSI(NNGV,IMV,JMV,KMV),P2(IMV,JMV)                       FLU00150
C    WILL USE IOSC1 TO BUILD FLUXES FOR TRANSMITTAL TO TIMDEP           FLU00160
      REWIND IOSC1                                                      FLU00170
      IF(IOPT.EQ.1)GO TO 200                                            FLU00180
      DO 100 K=1,KM                                                     FLU00190
      DO 100 NG=1,NNG                                                   FLU00200
      WRITE(IOSC1)((PSI(NG,I,J,K),I=1,IM),J=1,JM)                       FLU00210
  100 CONTINUE                                                          FLU00220
      REWIND IOSC1                                                      FLU00230
      GO TO 300                                                         FLU00240
  200 CONTINUE                                                          FLU00250
      K=0                                                               FLU00260
  210 K=K+1                                                             FLU00270
      ITEMP2=K-1                                                        FLU00280
      IF(ITEMP2.EQ.0)GO TO 230                                          FLU00290
      DO 220 ITEMP=1,ITEMP2                                             FLU00300
      READ(IOPO)                                                        FLU00310
  220 CONTINUE                                                          FLU00320
  230 DO 250 ITEMP=1,NNG                                                FLU00330
      READ(IOPO)P2                                                      FLU00340
      WRITE(IOSC1)P2                                                    FLU00350
      IF(ITEMP.EQ.NNG)GO TO 250                                         FLU00360
      DO 240 ITEMP3=1,KMX                                               FLU00370
      READ(IOPO)                                                        FLU00380
  240 CONTINUE                                                          FLU00390
  250 CONTINUE                                                          FLU00400
```

```
      REWIND IOPD                                                       FLU00410
      IF(K.LT.KM)GO TO 210                                              FLU00420
      REWIND IOSC1                                                      FLU00430
  300 RETURN                                                            FLU00440
      END                                                               FLU00450
```

```
      SUBROUTINE TIMDEP(V,XI,XIM,XNU,SIGF,SIGR,SIGT,SIGS,ALAM,BETA,XIP, TIM00010
     1X,Y,Z,HX,HY,HZ,IBP,JBP,KBP,DD1,DD2,DD3,DD4,DD5,DD6,DD7,VO,MMAP,NPRTIM00020
     2MP,PSI,P1,P2,P3,PSO,W,PO,W1,NNGV,NDGV,NTOGV,NDNSCV,NMATV,IMV,JMV,KTIM00030
     3MV,IRMV,JRMV,KRMV,NPRGV,NGXV)                                    TIM00040
      IMPLICIT REAL*8 (A-H,O-Z)                                         TIM00050
      INTEGER*2 MMAP,NPRMP                                              TIM00060
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   TIM00070
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    TIM00080
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,TIM00090
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     TIM00100
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    TIM00110
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    TIM00160
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           TIM00170
      COMMON/TIMINT/LASZON,ISTPCH,ILINCH,IPRSTP,MNSCH(5),MNLCH(5),      TIM00180
     1ISTEP,ICHHT                                                       TIM00190
      COMMON/TIMFLO/T,HT,HMIN,HMAX,TSTART,TEND,DELSFS(5,4),DELSRS(5,4), TIM00200
     1DELSTS(5,4),DELS1S(5,4),DELS2S(5,4),DELSFL(5,4),DELSRL(5,4),      TIM00210
     2DELSTL(5,4),DELS1L(5,4),DELS2L(5,4)                               TIM00220
      DIMENSION V(NNGV),XI(NNGV),XIM(NNGV),XNU(NMATV,NNGV),             TIM00230
     1SIGF(NMATV,NNGV),SIGR(NMATV,NNGV),SIGT(NMATV,NNGV),SIGS(NMATV,NNGVTIM00240
     2,NDNSCV),ALAM(NDGV),BETA(NDGV),XIP(NNGV,NDGV),X(IMV),Y(JMV),Z(KMV)TIM00250
     3,HX(IRMV),HY(JRMV),HZ(KRMV),IBP(IRMV),JBP(JRMV),KBP(KRMV),DD1(NPRGTIM00260
     4V,NNGV),DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),DD5(NPRGV,TIM00270
     5NNGV),DD6(NPRGV,NNGV),DD7(NPRGV,NGXV,NDNSCV),MMAP(IRMV,JRMV,KRMV),TIM00280
     6NPRMP(IMV,JMV,KMV),PSI(NTOGV,IMV,JMV,KMV),P1(NTOGV,IMV,JMV),      TIM00290
     7P2(NTOGV,IMV,JMV),P3(NTOGV,IMV,JMV),PSO(IMV,JMV,KMV),W(IMV,JMV,KMVTIM00300
     8),PO(IMV,JMV),W1(IMV,JMV),VO(NPRGV)                               TIM00310
      IF(IOPT.EQ.0) GO TO 100                                           TIM00320
      REWIND IOPO                                                       TIM00330
      REWIND IOPN                                                       TIM00340
      REWIND IOFO                                                       TIM00350
      REWIND IOFN                                                       TIM00360
  100 DO 105 NPR=1,NPRG                                                 TIM00365
      DO 105 NG=1,NNG                                                   TIM00370
      DD4(NPR,NG)=0.5D0*DD4(NPR,NG)                                     TIM00375
      DD5(NPR,NG)=DD5(NPR,NG)-DD4(NPR,NG)-XIM(NG)*DD6(NPR,NG)           TIM00380
```

```
  105 CONTINUE                                                          TIM00390
C   CALL DELAYS TO COMPUTE INITIAL DELAYED NEWTRON PRECURSOR DENSITIES  TIM00400
C   AND READ FLUXES FROM IOSC1                                          TIM00410
      CALL DELAYS(ALAM,BETA,XIP,DD6,VO,NPRMP,PSI,P2,PSO,PO,NNG,NDG,NTOG,TIM00420
     1NMAT,IM,JM,KM,NPRG)                                               TIM00430
      DO 120 ND=1,NDG                                                   TIM00431
      DO 110 NG=1,NNG                                                   TIM00432
  110 XIP(NG,ND)=XIP(NG,ND)*ALAM(ND)                                    TIM00433
  120 ALAM(ND)=ALAM(ND)/2.0D0                                           TIM00434
C   ZERO FREQUENCY VECTOR                                               TIM00440
      DO 130 K=1,KM                                                     TIM00450
      DO 130 J=1,JM                                                     TIM00460
      DO 130 I=1,IM                                                     TIM00470
  130 W(I,J,K)=0.0D0                                                    TIM00480
      TSTART=0.0D0                                                      TIM00490
      ISTEP=0                                                           TIM00500
C   START LOOP HERE OVER TIME ZONES BY CALLING TIMINP                   TIM00510
  200 CALL TIMINP                                                       TIM00520
      NFLAG1=1                                                          TIM00530
      IF(ISTPCH.GT.0)CALL CHANGE(XIM,XNU,SIGF,SIGR,SIGT,SIGS,HX,HY,HZ,  TIM00540
     1IBP,JBP,KBP,DD1,DD2,DD3,DD4,DD5,DD6,DD7,MMAP,NNG,NDNSCT,NMAT,IM,JMTIM00550
     2,KM,IRM,JRM,KRM,NPRG,NFLAG1,NGX)                                  TIM00560
      T=TSTART                                                          TIM00570
      HT=HMIN                                                           TIM00580
      NFLAG2=1                                                          TIM00590
      IF(ISTEP.EQ.0)CALL TIMOUT(PSI,P2,W,W1,NTOG,IM,JM,KM,NFLAG2)       TIM00600
  210 IF(IOPT.EQ.1)GO TO 230                                            TIM00610
      CALL STEPAO(V,XIM,ALAM,BETA,XIP,X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5TIM00620
     1,DD6,DD7,VO,NPRMP,PSI,W,NNG,NDG,NTOG,NDNSCT,IM,JM,KM,IRM,JRM,KRM, TIM00630
     2NPRG,NGX)                                                         TIM00640
      CALL STEPBO(V,XIM,ALAM,BETA,XIP,X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,DD5TIM00650
     1,DD6,DD7,VO,NPRMP,PSI,W,NNG,NDG,NTOG,NDNSCT,IM,JM,KM,IRM,JRM,KRM, TIM00660
     2NPRG,NGX)                                                         TIM00670
      CALL FREQO(PSI,PSO,W,NTOG,IM,JM,KM)                               TIM00680
      DO 220 K=1,KM                                                     TIM00690
      DO 220 J=1,JM                                                     TIM00700
```

```
      DO 220 I=1,IM                                                     TIM00710
  220 PSO(I,J,K)=PSI(NTG,I,J,K)                                         TIM00720
      GO TO 250                                                         TIM00730
  230 CONTINUE                                                          **TEMP**
C 230 CALL STEPA1(V,XIM,ALAM,BETA,XIP,X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,    TIM00740
C    1DD5,DD6,DD7,NPRMP,P1,P2,P3,W,PO,W1,NNG,NDG,NTOG,NDNSCT,IM,JM,KM,  TIM00750
C    2IRM,JRM,KRM,NPRG)                                                 TIM00760
C     CALL STEPB1(V,XIM,ALAM,BETA,XIP,X,Y,Z,HX,HY,HZ,DD1,DD2,DD3,DD4,    TIM00770
C    1DD5,DD6,DD7,NPRMP,P1,P2,P3,W,PO,W1,NNG,NDG,NTOG,NDNSCT,IM,JM,KM,  TIM00780
C    2IRM,JRM,KRM,NPRG)                                                 TIM00790
C     CALL FREQ1(P2,PO,W,W1,NTOG,IM,JM,KM)                              TIM00800
  250 T=T+2.0D0*HT                                                      TIM00810
      ISTEP=ISTEP+1                                                     TIM00820
      NFLAG1=2                                                          TIM00830
      NFLAG2=0                                                          TIM00840
      IF(ILINCH.GT.0)CALL CHANGE(XIM,XNU,SIGF,SIGR,SIGT,SIGS,HX,HY,HZ,  TIM00850
     1IBP,JBP,KBP,DD1,DD2,DD3,DD4,DD5,DD6,DD7,MMAP,NNG,NDNSCT,NMAT,IM,JMTIM00860
     2,KM,IRM,JRM,KRM,NPRG,NFLAG1,NGX)                                  TIM00870
      IF(DABS(T-TEND).LT.1.0D-10)NFLAG2=1                               TIM00880
      IF(NFLAG2.EQ.1.OR.MOD(ISTEP,IPRSTP).EQ.0)CALL TIMOUT(PSI,P2,W,W1, TIM00890
     1NTOG,IM,JM,KM,NFLAG2)                                             TIM00900
      IF(ICHHT.EQ.1)CALL TALTER                                         TIM00910
      CALL ETIMEF(TEMP)                                                 TIM00920
      IF(TEMP.LT.TIME)GO TO 270                                         TIM00930
      NFLAG2=2                                                          TIM00940
      LASZON=1                                                          TIM00950
      CALL TIMOUT(PSI,P2,W,W1,NTOG,IM,JM,KM,NFLAG2)                     TIM00960
      GO TO 280                                                         TIM00970
  270 IF(NFLAG2.EQ.0)GO TO 210                                          TIM00980
      TSTART=T                                                          TIM00990
      IF(LASZON.GT.0)GO TO 200                                          TIM01000
  280 IF(IOPT.EQ.0)GO TO 300                                            TIM01010
      REWIND IOFO                                                       TIM01020
      REWIND IOFN                                                       TIM01030
      REWIND IOPO                                                       TIM01040
      REWIND IOPN                                                       TIM01050
```

```
      REWIND IOSC1                                                      TIM01060
      REWIND IOSC2                                                      TIM01070
  300 RETURN                                                            TIM01080
      END                                                               TIM01090
```

```
      SUBROUTINE TIMINP
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 MMAP,NPRMP
      COMMON/INT3/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT
      COMMON/TIMINT/LASZON,ISTPCH,ILINCH,IPRSTP,MNSCH(5),MNLCH(5),
     1ISTEP,ICHHT
      COMMON/TIMFLO/T,HT,HMIN,HMAX,TSTART,TEND,DELSFS(5,4),DELSRS(5,4),
     1DELSTS(5,4),DELS1S(5,4),DELS2S(5,4),DELSFL(5,4),DELSRL(5,4),
     2DELSTL(5,4),DELS1L(5,4),DELS2L(5,4)
C    READ IN FIRST TIME ZONE DESCRIPTION CARD (CARD TYPE 13)
  100 READ(5,1000)LASZON,ISTPCH,ILINCH,IPRSTP,ICHHT,IFLOUT,HMIN,HMAX,TEN
     1D
 1000 FORMAT(6I5,3D12.5)
      IF(ISTEP.GT.0)WRITE(6,1010)
 1010 FORMAT(1H1,//)
      IF(LASZON.GT.0)GO TO 110
      LTMZON=LTMZON+1
      GO TO 120
  110 LTMZON=LASZON
  120 WRITE(6,1020)LTMZON
 1020 FORMAT(1H0,//,15X,'EDITED INPUT FOR TIME ZONE',I3,//)
      WRITE(6,1030)LASZON,ISTPCH,ILINCH,IPRSTP,ICHHT,IFLOUT,HMIN,HMAX,TE
     1ND
C    IF ISTPCH GT 0, READ IN STEP CHANGE INFORMATION
      IF(ISTPCH.EQ.0)GO TO 140
      DO 130 MN=1,ISTPCH
      DO 130 NG=1,NNG
      READ(5,1040)MNSCH(MN),DELSFS(MN,NG),DELSRS(MN,NG),DELSTS(MN,NG),
     1DELS1S(MN,NG),DELS2S(MN,NG)
      WRITE(6,1050)MNSCH(MN),DELSFS(MN,NG),DELSRS(MN,NG),DELSTS(MN,NG),
```

```
     1DELS1S(MN,NG),DELS2S(MN,NG)                                        TIM00390
 130 CONTINUE                                                            TIM00400
1030 FORMAT(11X,6I5,3D12.5)                                              TIM00410
1040 FORMAT(I5,5X,5D12.5)                                                TIM00420
1050 FORMAT(11X,I5,5X,5D12.5)                                            TIM00430
 140 IF(ILINCH.EQ.0)GO TO 160                                            TIM00440
     DO 150 MN=1,ILINCH                                                  TIM00450
     DO 150 NG=1,NNG                                                     TIM00460
     READ(5,1040)MNLCH(MN),DELSFL(MN,NG),DELSRL(MN,NG),DELSTL(MN,NG),    TIM00470
    1DELS1L(MN,NG),DELS2L(MN,NG)                                         TIM00480
     WRITE(6,1050)MNLCH(MN),DELSFL(MN,NG),DELSRL(MN,NG),DELSTL(MN,NG),   TIM00490
    1DELS1L(MN,NG),DELS2L(MN,NG)                                         TIM00500
 150 CONTINUE                                                            TIM00510
C    NOW PRINT OUT EDITED INFORMATION                                    TIM00520
 160 WRITE(6,1060)HMIN,HMAX,TEND                                         TIM00530
1060 FORMAT(1H0,10X,'MIN. TIME STEP(SEC)= ',D12.6,'  MAX. TIME STEP(SECTIM00540
    1)= ',D12.6,'  ZONE END TIME(SEC)= ',D12.6)                          TIM00550
     IF(ISTPCH.EQ.0)GO TO 180                                            TIM00560
     WRITE(6,1070)ISTPCH                                                 TIM00570
1070 FORMAT(1H0,10X,'STEP CHANGES IN',I2,' MATERIALS IN THIS TIME ZONE TIM00580
    1')                                                                  TIM00590
     WRITE(6,1080)                                                       TIM00600
1080 FORMAT(1H0,55X,'TOTAL CHANGE (IN CM-1) IN CROSS-SECTIONS',/11X,     TIM00610
    1'MATERIAL',4X,'GROUP',70X,'SCATTERING',/,35X,'FISSION',10X,         TIM00620
    2'ABSORPTION',8X,'TRANSPORT',10X,'G TO G+1',10X,'G TO G+2',/)        TIM00630
     DO 170 MN=1,ISTPCH                                                  TIM00640
     DO 170 NG=1,NNG                                                     TIM00650
     WRITE(6,1090)MNSCH(MN),NG,DELSFS(MN,NG),DELSRS(MN,NG),DELSTS(MN,NGTIM00660
    1),DELS1S(MN,NG),DELS2S(MN,NG)                                       TIM00670
 170 CONTINUE                                                            TIM00680
1090 FORMAT(1H ,14X,I2,7X,I2,2X,5(4X,D14.7))                             TIM00690
 180 IF(ILINCH.EQ.0)GO TO 200                                            TIM00700
     WRITE(6,1100)ILINCH                                                 TIM00710
1100 FORMAT(1H0,10X,'RAMP CHANGES IN ',I2,' MATERIALS IN THIS TIME ZONETIM00720
    1')                                                                  TIM00730
     WRITE(6,1080)                                                       TIM00740
```

```
      DO 190 MN=1,ILINCH
      DO 190 NG=1,NNG
      WRITE(6,1090)MNLCH(MN),NG,DELSFL(MN,NG),DELSRL(MN,NG),DELSTL(MN,NG
     1),DELS1L(MN,NG),DELS2L(MN,NG)
  190 CONTINUE
  200 WRITE(6,1110)
 1110 FORMAT(1H0,//,10X,'BEGIN TIME-DEPENDENT CALCULATION FOR THIS ZONE'
     1)
      RETURN
      END
```

```
      SUBROUTINE TALTER                                                 TAL00010
      IMPLICIT REAL*8 (A-H,O-Z)                                         TAL00020
      INTEGER*2 MMAP,NPRMP                                              TAL00030
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM, JM,KM,IRM,JRM,KRM,NLBC,  TAL00040
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    TAL00050
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,TAL00060
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     TAL00070
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    TAL00080
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    TAL00130
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           TAL00140
      COMMON/TIMINT/LASZON,ISTPCH,ILINCH,IPRSTP,MNSCH(5),MNLCH(5),      TAL00150
     1ISTEP,ICHHT                                                       TAL00160
      COMMON/TIMFLO/T,HT,HMIN,HMAX,TSTART,TEND,DELSFS(5,4),DELSRS(5,4), TAL00170
     1DELSTS(5,4),DELS1S(5,4),DELS2S(5,4),DELSFL(5,4),DELSRL(5,4),      TAL00180
     2DELSTL(5,4),DELS1L(5,4),DELS2L(5,4)                               TAL00190
C   THE FOLLOWING LOGIC ASSURES THAT HT IS AN INTEGER MULTIPLE OF TIME  TAL00200
C   ZONE LENGTH                                                         TAL00210
      TEMP5=(TEND-TSTART)/(2.0D0*HT)                                    TAL00220
      ITEMP5=TEMP5                                                      TAL00230
      TEMP6=ITEMP5                                                      TAL00240
      IF((TEMP5-TEMP6).LT.1.0D-11)GO TO 110                             TAL00250
 1000 FORMAT(1H0,15X,'*****INPUT HMIN (=HT) IS NOT AN INTEGER MULTIPLE OTAL00260
     1F TIME ZONE LENGTH*****')                                         TAL00270
      HT=(TEND-TSTART-2.0D0*HT)/(TEMP6-1.0D0)                           TAL00280
      WRITE(6,1000)                                                     TAL00290
      WRITE(6,1010)HT                                                   TAL00300
 1010 FORMAT(1H ,15X,'HT HAS BEEN CHANGED TO ',D20.13,' SECONDS AND WILLTAL00310
     1 BE HELD FIXED AT THAT VALUE')                                    TAL00320
  110 ICHHT=0                                                           TAL00330
      RETURN                                                            TAL00340
      END                                                               TAL00350
```

```
      SUBROUTINE DSIMQ(A,B,N,KS)
C   THIS SUBROUTINE HAS BEEN TAKEN FROM THE IBM SCIENTIFIC
C   SUBROUTINE PACKAGE AND CONVERTED TO DOUBLE PRECISION
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(1),B(1)
C
C        FORWARD SOLUTION
C
      TOL=0.0
      KS=0
      JJ=-N
      DO 65 J=1,N
      JY=J+1
      JJ=JJ+N+1
      BIGA=0
      IT=JJ-J
      DO 30 I=J,N
C
C        SEARCH FOR MAXIMUM COEFFICIENT IN COLUMN
C
      IJ=IT+I
      IF(DABS(BIGA)-DABS(A(IJ))) 20,30,30
   20 BIGA=A(IJ)
      IMAX=I
   30 CONTINUE
C
C        TEST FOR PIVOT LESS THAN TOLERANCE (SINGULAR MATRIX)
C
      IF(DABS(BIGA)-TOL) 35,35,40
   35 KS=1
      RETURN
C
C        INTERCHANGE ROWS IF NECESSARY
C
   40 I1=J+N*(J-2)
      IT=IMAX-J
```

```
      DO 50 K=J,N
      I1=I1+N
      I2=I1+IT
      SAVE=A(I1)
      A(I1)=A(I2)
      A(I2)=SAVE
C
C        DIVIDE EQUATION BY LEADING COEFFICIENT
C
   50 A(I1)=A(I1)/BIGA
      SAVE=B(IMAX)
      B(IMAX)=B(J)
      B(J)=SAVE/BIGA
C
C        ELIMINATE NEXT VARIABLE
C
      IF(J-N) 55,70,55
   55 IQS=N*(J-1)
      DO 65 IX=JY,N
      IXJ=IQS+IX
      IT=J-IX
      DO 60 JX=JY,N
      IXJX=N*(JX-1)+IX
      JJX=IXJX+IT
   60 A(IXJX)=A(IXJX)-(A(IXJ)*A(JJX))
   65 B(IX)=B(IX)-(B(J)*A(IXJ))
C
C        BACK SOLUTION
C
   70 NY=N-1
      IT=N*N
      DO 80 J=1,NY
      IA=IT-J
      IB=N-J
      IC=N
      DO 80 K=1,J
```

```
      B(IB)=B(IB)-A(IA)*B(IC)                                         SI00730
      IA=IA-N                                                         SI00740
   80 IC=IC-1                                                         SI00750
      RETURN                                                          SI00760
      END                                                             SI00770
```

```
      SUBROUTINE TIMOUT(PSI,P2,W,W1,NTOGV,IMV,JMV,KMV,NFLAG2)           TIM00010
      IMPLICIT REAL*8 (A-H,O-Z)                                         TIM00020
      INTEGER*2 MMAP,NPRMP                                              TIM00030
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   TIM00040
     1NFBC,NBBC,NONSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    TIM00050
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,TIM00060
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     TIM00070
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    TIM00080
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    TIM00130
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           TIM00140
      COMMON/TIMINT/LASZON,ISTPCH,ILINCH,IPRSTP,MNSCH(5),MNLCH(5),      TIM00150
     1ISTEP,ICHHT                                                       TIM00160
      COMMON/TIMFLO/T,HT,HMIN,HMAX,TSTART,TEND,DELSFS(5,4),DELSRS(5,4), TIM00170
     1DELSTS(5,4),DELS1S(5,4),DELS2S(5,4),DELSFL(5,4),DELSRL(5,4),      TIM00180
     2DELSTL(5,4),DELS1L(5,4),DELS2L(5,4)                               TIM00190
      DIMENSION PSI(NTOGV,IMV,JMV,KMV),P2(NTOGV,IMV,JMV),W(IMV,JMV,KMV),TIM00200
     1W1(IMV,JMV)                                                       TIM00210
      CALL ETIMEF(TEMP)                                                 TIM00220
      WRITE(6,1000)                                                     TIM00230
 1000 FORMAT(1H1,//)                                                    TIM00240
      IF(ISTEP.GT.0)GO TO 100                                           TIM00250
      WRITE(6,1010)(ITITLE(I),I=1,20)                                   TIM00260
 1010 FORMAT(1H ,10X,'INITIAL FLUXES FOR THE PROBLEM ',20A4)            TIM00270
      ISSAVE=ISTEP                                                      TIM00280
  100 WRITE(6,1020)ISTEP,T,HT,TEMP                                      TIM00290
 1020 FORMAT(1H0,5X,'STEP NUMBER',I4,2X,'TRANSIENT TIME(SEC)=',1PD14.7  TIM00300
     1,2X,'1/2 TIME STEP(SEC)= ',1PD14.7,2X,'ELAPSED CPU TIME(MIN)=',   TIM00310
     20PF10.4)                                                          TIM00320
      IF(ISTEP.EQ.0)GO TO 230                                           TIM00330
C   WRITE OUT FREQUENCIES AT TEST POINTS                                TIM00340
      WRITE(6,1030)                                                     TIM00350
 1030 FORMAT(1H0,/,15X,'FREQUENCIES AT TEST POINTS',/)                  TIM00360
      DO 130 K=1,NZTP                                                   TIM00370
      IF(K.GT.1)GO TO 110                                               TIM00380
      WRITE(6,1040)(IXTP(I),I=1,NXTP)                                   TIM00390
 1040 FORMAT(1H ,24X,'J / I',7X,5(I3,15X))                              TIM00400
```

```
  110 WRITE(6,1050)IZTP(K)                                              TIM00410
 1050 FORMAT(1H0,12X,'PLANE ',I2)                                       TIM00420
      DO 120 JJ=1,NYTP                                                  TIM00430
      J=NYTP+1-JJ                                                       TIM00440
      WRITE(6,1060)IYTP(J),(W(IXTP(I),IYTP(J),IZTP(K)),I=1,NXTP)        TIM00450
  120 CONTINUE                                                          TIM00460
 1060 FORMAT(1H ,22X,I3,2X,5(4X,1PD14.7))                               TIM00470
  130 CONTINUE                                                          TIM00480
      IF(IFLOUT.GT.0.AND.NFLAG2.GT.0)GO TO 220                          TIM00490
C   GO HERE FOR WRITING OUT FLUXES AT TEST POINTS ONLY                  TIM00500
      IF((NZTP*(NYTP+2)).GT.26)WRITE(6,1000)                            TIM00510
      WRITE(6,1070)                                                     TIM00520
 1070 FORMAT(1H0,/,15X,'FLUXES AT TEST POINTS',/)                       TIM00530
      LINECT=(NZTP*(NYTP+2))+14                                         TIM00540
      IF((NZTP*(NYTP+2)).GT.26)LINECT=10                                TIM00550
C   IF IOPT=1,ASSUME NEW FLUXES ON  IOPD AND THAT IOPD IS REWOUND       TIM00560
      KS=1                                                              TIM00570
      DO 210 K=1,NZTP                                                   TIM00580
      IF(K.GT.1)GO TO 140                                               TIM00590
      WRITE(6,1040)(IXTP(I),I=1,NXTP)                                   TIM00600
  140 IF(IOPT.EQ.0)GO TO 170                                            TIM00610
      KD=IZTP(K)-KS                                                     TIM00620
      IF(KD.EQ.0)GO TO 160                                              TIM00630
      DO 150 ITEMP3=1,KD                                                TIM00640
      READ(IOPD)                                                        TIM00650
  150 CONTINUE                                                          TIM00660
  160 READ(IOPD)P2                                                      TIM00670
  170 DO 200 NG=1,NNG                                                   TIM00680
      ND=NG-NNG                                                         TIM00690
      IF(NG.LE.NNG)WRITE(6,1080)IZTP(K),NG                              TIM00700
      IF(NG.GT.NNG)WRITE(6,1090)IZTP(K),ND                              TIM00710
 1080 FORMAT(1H0,12X,'PLANE ',I2,' , NEUTRON GROUP ',I2)                TIM00720
 1090 FORMAT(1H0,12X,'PLANE ',I2,' , PRECURSOR GROUP ',I2)              TIM00730
      DO 190 JJ=1,NYTP                                                  TIM00740
      J=NYTP+1-JJ                                                       TIM00750
      IF(IOPT.EQ.0)GO TO 180                                            TIM00760
```

```
      WRITE(6,1060)IYTP(J),(P2(NG,IXTP(I),IYTP(J)),I=1,NXTP)
      GO TO 190
  180 WRITE(6,1050)IYTP(J),(PSI(NG,IXTP(I),IYTP(J),IZTP(K)),I=1,NXTP)
  190 CONTINUE
      LINECT=LINECT+NYTP+2
      IF((LINECT+NYTP+2).LE.60)GO TO 200
      WRITE(6,1000)
      WRITE(6,1070)
      WRITE(6,1040)(IXTP(I),I=1,NXTP)
      LINECT=7
  200 CONTINUE
      KS=IZTP(K)
  210 CONTINUE
      GO TO 290
C    BRANCH HERE FOR COMPLETE FLUX DUMP
  220 WRITE(6,1000)
      WRITE(6,1100)(ITITLE(I),I=1,20)
 1100 FORMAT(1H0,10X,'FLUXES FOR THE PROBLEM',20A4)
  230 DO 280 K=1,KM
      IF(IOPT.EQ.1)READ(IOPO)P2
      DO 280 NG=1,NTOG
      ND=NG-NNG
      IF(K.GT.1.OR.NG.GT.1)WRITE(6,1110)
 1110 FORMAT(1H1,/)
      IF(NG.LE.NNG)WRITE(6,1120)K,NG
      IF(NG.GT.NNG)WRITE(6,1130)K,ND
 1120 FORMAT(1H0,10X,'NEUTRON FLUXES FOR PLANE ',I2,' , GROUP ',I2)
 1130 FORMAT(1H0,10X,'PRECURSOR CONC. FOR PLANE ',I2,' , GROUP ',I2)
      JMS=1
      JME=JM
      IF(JM.GT.50)JME=50
      ITEMP2=50/JME
      ITEMP4=ITEMP2
      DO 270 I=1,IM,10
      IS=I
      IE=I+9
```

```
      IF(IE.GT.IM)IE=IM
      IF((I-1)/10.LT.ITEMP4)GO TO 240
      WRITE(6,1110)
      ITEMP4=ITEMP4+ITEMP2
  240 WRITE(6,1140)(ITEMP3,ITEMP3=IS,IE)
 1140 FORMAT(1H0,3X,'J / I',2X,I7,9I12)
      WRITE(6,1150)
 1150 FORMAT(1H ,3X)
      DO 260 ITEMP3=JMS,JME
      J=JME+1-ITEMP3
      IF(IOPT.EQ.1)GO TO 250
      WRITE(6,1160)J,(PSI(NG,II,J,K),II=IS,IE)
 1160 FORMAT(1H ,2X,I2,6X,1P10D12.5)
      GO TO 260
  250 WRITE(6,1160)J,(P2(NG,II,J),II=IS,IE)
  260 CONTINUE
      IF(JME.GE.JM)GO TO 270
      JMS=JME+1
      JME=JMS+49
      IF(JME.GT.JM)JME=JM
      WRITE(6,1110)
      GO TO 240
  270 CONTINUE
  280 CONTINUE
      CALL ETIMEF(TEMP)
      WRITE(6,1180)TEMP
 1180 FORMAT(1H0,10X,'FLUX PRINTOUT COMPLETED, ELAPSED TIME(MIN) = ',F10
     1.4)
      IF(NFLAG2.EQ.2)WRITE(6,1170)
 1170 FORMAT(1H1,10X,'HAVE USED ALLOTTED CPU TIME')
  290 CONTINUE
      IF(IOPT.EQ.1)REWIND IOPO
      RETURN
      END
```

```
      SUBROUTINE CHANGE(XIM,XNU,SIGF,SIGR,SIGT,SIGS,HX,HY,HZ,IBP,JBP,KBPSET00010
     1,DD1,DD2,DD3,DD4,DD5,DD6,DD7,MMAP,NNGV,NDNSCV,NMATV,IMV,JMV,KMV,  SET00020
     2IRMV,JRMV,KRMV,NPRGV,NFLAG1,NGXV)                                  SET00030
      IMPLICIT REAL*8 (A-H,O-Z)                                          SET00040
      INTEGER*2 MMAP,NPRMP                                               SET00050
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,    SET00060
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),     SET00070
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,SET00080
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,      SET00090
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                     SET00100
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,     SET00150
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT             SET00160
      COMMON/TIMINT/LASZON,ISTPCH,ILINCH,IPRSTP,MNSCH(5),MNLCH(5),       SET00170
     1ISTEP,ICHHT                                                        SET00180
      COMMON/TIMFLO/T,HT,HMIN,HMAX,TSTART,TEND,DELSFS(5,4),DELSRS(5,4),  SET00190
     1DELSTS(5,4),DELS1S(5,4),DELS2S(5,4),DELSFL(5,4),DELSRL(5,4),       SET00200
     2DELSTL(5,4),DELS1L(5,4),DELS2L(5,4)                                SET00210
      DIMENSION XIM(NNGV),XNU(NMATV,NNGV),SIGF(NMATV,NNGV),SIGR(NMATV,   SET00220
     1NNGV),SIGT(NMATV,NNGV),SIGS(NMATV,NNGV,NDNSCV),HX(IRMV),HY(JRMV),  SET00230
     2HZ(KRMV),IBP(IRMV),JBP(JRMV),KBP(KRMV),DD1(NPRGV,NNGV),DD2(NPRGV,  SET00240
     3NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),DD5(NPRGV,NNGV),DD6(NPRGV,   SET00250
     4NNGV),DD7(NPRGV,NGXV,NDNSCV),MMAP(IRMV,JRMV,KRMV)                  SET00260
      DIMENSION HD(6),MN(8)                                              SET00270
      TEMP=1.2D1                                                         SET00280
      TEMP1=8.0D0                                                        SET00290
C    FIRST ALTER CROSS SECTIONS                                          SET00300
      IF(NFLAG1.EQ.2)GO TO 110                                           SET00310
      ITEMP1=ISTPCH                                                      SET00320
      TEMP2=1.0D0                                                        SET00330
      GO TO 120                                                          SET00340
  110 ITEMP1=ILINCH                                                      SET00350
      TEMP2=2.0D0*HT/(TEND-TSTART)                                       SET00360
  120 DO 600 ITEMP2=1,ITEMP1                                             SET00370
      IF(NFLAG1.EQ.2)GO TO 150                                           SET00380
  130 NM=MNSCH(ITEMP2)                                                   SET00390
      MM=ITEMP2                                                          SET00395
```

```
      DO 140 NG=1,NNG                                                   SET00400
      SIGF(NM,NG)=SIGF(NM,NG)+DELSFS(MM,NG)                             SET00410
      SIGR(NM,NG)=SIGR(NM,NG)+DELSRS(MM,NG)+DELS1S(MM,NG)+DELS2S(MM,NG) SET00420
      SIGT(NM,NG)=SIGT(NM,NG)+DELSTS(MM,NG)                             SET00430
      SIGS(NM,NG,1)=SIGS(NM,NG,1)+DELS1S(MM,NG)                         SET00440
      IF(NDNSCT.LT.2)GO TO 140                                          SET00450
      SIGS(NM,NG,2)=SIGS(NM,NG,2)+DELS2S(MM,NG)                         SET00460
  140 CONTINUE                                                          SET00470
      GO TO 170                                                         SET00480
  150 NM=MNLCH(ITEMP2)                                                  SET00490
      MM=ITEMP2                                                         SET00495
      DO 160 NG=1,NNG                                                   SET00500
      SIGF(NM,NG)=SIGF(NM,NG)+TEMP2*DELSFL(MM,NG)                       SET00510
      SIGR(NM,NG)=SIGR(NM,NG)+TEMP2*(DELSRL(MM,NG)+DELS1L(MM,NG))       SET00520
      SIGT(NM,NG)=SIGT(NM,NG)+TEMP2*DELSTL(MM,NG)                       SET00530
      SIGS(NM,NG,1)=SIGS(NM,NG,1)+TEMP2*DELS1L(MM,NG)                   SET00540
      IF(NDNSCT.LT.2)GO TO 160                                          SET00550
      SIGR(NM,NG)=SIGR(NM,NG)+TEMP2*DELS2L(MM,NG)                       SET00560
      SIGS(NM,NG,2)=SIGS(NM,NG,2)+TEMP2*DELS2L(MM,NG)                   SET00570
  160 CONTINUE                                                          SET00580
C   LOOP OVER MATERIAL REGIONS, CHANGING COEFFICIENTS WHENEVER MMAP(IR,JSET00590
C   R,KR)=NM                                                            SET00600
  170 DO 550 KR=1,KRM                                                   SET00610
      DO 540 JR=1,JRM                                                   SET00620
      DO 530 IR=1,IRM                                                   SET00630
      IF(MMAP(IR,JR,KR).NE.NM)GO TO 530                                 SET00640
C   HOMOGENEOUS REGION                                                  SET00650
      NPR=4*IRM*JRM*(2*KR-1)+2*IRM*(2*JR-1)+2*IR                        SET00660
      ITEMP5=1                                                          SET00670
      NPRP=NPR                                                          SET00680
      HD(1)=HZ(KR)                                                      SET00690
      HD(2)=HD(1)                                                       SET00700
      HD(3)=HY(JR)                                                      SET00710
      HD(4)=HD(3)                                                       SET00720
      HD(5)=HX(IR)                                                      SET00730
      HD(6)=HD(5)                                                       SET00740
```

```
      DO 180 ITEMP4=1,8
      MN(ITEMP4)=MMAP(IR,JR,KR)
  180 CONTINUE
      GO TO 500
C   LOWER LEFT EDGE
  200 NPRP=NPR-4*IRM*JRM-1
      ITEMP5=2
      HD(1)=HZ(KR)
      HD(2)=HZ(KR-1)
      IF(KR.EQ.1)HD(2)=HD(1)
      HD(5)=HX(IR)
      HD(6)=HX(IR-1)
      IF(IR.EQ.1)HD(6)=HD(5)
      MN(1)=MMAP(IR,JR,KR-1)
      IF(KR.EQ.1)MN(1)=MN(5)
      MN(4)=MN(1)
      MN(6)=MMAP(IR-1,JR,KR)
      IF(IR.EQ.1)MN(6)=MN(5)
      MN(7)=MN(6)
      MN(2)=MMAP(IR-1,JR,KR-1)
      IF(KR.EQ.1)MN(2)=MN(6)
      IF(IR.EQ.1)MN(2)=MN(1)
      MN(3)=MN(2)
      GO TO 500
C   LEFT SIDE
  210 NPRP=NPR-1
      ITEMP5=3
      HD(2)=HD(1)
      MN(4)=MN(8)
      MN(1)=MN(5)
      MN(2)=MN(6)
      MN(3)=MN(7)
      GO TO 500
C   LEFT FRONT EDGE
  220 NPRP=NPR-2*IRM-1
      ITEMP5=4
```

```
      MN(8)=MMAP(IR,JR-1,KR)                                            SET01110
      IF(JR.EQ.1)MN(8)=MN(5)                                            SET01120
      MN(4)=MN(8)                                                       SET01130
      MN(7)=MMAP(IR-1,JR-1,KR)                                          SET01140
      IF(IR.EQ.1)MN(7)=MN(8)                                            SET01150
      IF(JR.EQ.1)MN(7)=MN(6)                                            SET01160
      MN(3)=MN(7)                                                       SET01170
      HD(4)=HY(JR-1)                                                    SET01180
      IF(JR.EQ.1)HD(4)=HD(3)                                            SET01190
      GO TO 500                                                         SET01200
C    LOWER FRONT EDGE                                                   SET01210
  230 NPRP=NPR-4*IRM*JRM-2*IRM                                          SET01220
      ITEMP5=5                                                          SET01230
      HD(2)=HZ(KR-1)                                                    SET01240
      IF(KR.EQ.1)HD(2)=HD(1)                                            SET01250
      HD(6)=HD(5)                                                       SET01260
      MN(6)=MN(5)                                                       SET01270
      MN(1)=MMAP(IR,JR,KR-1)                                            SET01280
      IF(KR.EQ.1)MN(1)=MN(5)                                            SET01290
      MN(2)=MN(1)                                                       SET01300
      MN(7)=MN(8)                                                       SET01310
      MN(4)=MMAP(IR,JR-1,KR-1)                                          SET01320
      IF(KR.EQ.1)MN(4)=MN(8)                                            SET01330
      IF(JR.EQ.1)MN(4)=MN(1)                                            SET01340
      MN(3)=MN(4)                                                       SET01350
      GO TO 500                                                         SET01360
C    LOWER FRONT LEFT CORNER                                            SET01370
  240 NPRP=NPR-4*IRM*JRM-2*IRM-1                                        SET01380
      ITEMP5=6                                                          SET01390
      HD(6)=HX(IR-1)                                                    SET01400
      IF(IR.EQ.1)HD(6)=HD(5)                                            SET01410
      MN(6)=MMAP(IR-1,JR,KR)                                            SET01420
      IF(IR.EQ.1)MN(6)=MN(5)                                            SET01430
      MN(2)=MMAP(IR-1,JR,KR-1)                                          SET01440
      IF(IR.EQ.1)MN(2)=MN(1)                                            SET01450
      IF(KR.EQ.1)MN(2)=MN(6)                                            SET01460
```

```
      MN(7)=MMAP(IR-1,JR-1,KR)
      IF(IR.EQ.1)MN(7)=MN(8)
      IF(JR.EQ.1)MN(7)=MN(6)
      MN(3)=MMAP(IR-1,JR-1,KR-1)
      IF(IR.EQ.1)MN(3)=MN(4)
      IF(JR.EQ.1)MN(3)=MN(2)
      IF(KR.EQ.1)MN(3)=MN(7)
      GO TO 500
C   FRONT SIDE
  250 NPRP=NPR-2*IRM
      ITEMP5=7
      HD(2)=HD(1)
      HD(6)=HD(5)
      MN(4)=MN(8)
      MN(7)=MN(8)
      MN(3)=MN(8)
      MN(6)=MN(5)
      MN(1)=MN(5)
      MN(2)=MN(5)
      GO TO 500
C   BOTTOM SIDE
  260 NPRP=NPR-4*IRM*JRM
      ITEMP5=8
      HD(2)=HZ(KR-1)
      IF(KR.EQ.1)HD(2)=HD(1)
      HD(4)=HD(3)
      MN(8)=MN(5)
      MN(7)=MN(6)
      MN(1)=MMAP(IR,JR,KR-1)
      IF(KR.EQ.1)MN(1)=MN(5)
      MN(2)=MN(1)
      MN(3)=MN(1)
      MN(4)=MN(1)
      GO TO 500
C   BOTTOM RIGHT EDGE (9)
  270 IF(IR.EQ.IRM)GO TO 360
```

```
      NPRP=NPR-4*IRM*JRM+1                                          SET01830
      ITEMP5=9                                                      SET01840
      HD(5)=HX(IR+1)                                                SET01850
      MN(5)=MMAP(IR+1,JR,KR)                                        SET01860
      MN(8)=MN(5)                                                   SET01870
      MN(1)=MMAP(IR+1,JR,KR-1)                                      SET01880
      IF(KR.EQ.1)MN(1)=MN(5)                                        SET01890
      MN(4)=MN(1)                                                   SET01900
      GO TO 500                                                     SET01910
C     FRONT BOTTOM RIGHT CORNER (10)                                SET01920
  280 NPRP=NPR-4*IRM*JRM-2*IRM+1                                    SET01930
      ITEMP5=10                                                     SET01940
      IF(JR.EQ.1)GO TO 285                                          SET01950
      HD(4)=HY(JR-1)                                                SET01960
      MN(7)=MMAP(IR,JR-1,KR)                                        SET01970
      MN(3)=MMAP(IR,JR-1,KR-1)                                      SET01980
      MN(8)=MMAP(IR+1,JR-1,KR)                                      SET01990
      MN(4)=MMAP(IR+1,JR-1,KR-1)                                    SET02000
      IF(KR.NE.1)GO TO 285                                          SET02010
      MN(3)=MN(7)                                                   SET02020
      MN(4)=MN(8)                                                   SET02030
  285 GO TO 500                                                     SET02040
C     FRONT RIGHT EDGE                                              SET02050
  290 NPRP=NPR-2*IRM+1                                              SET02060
      HD(2)=HD(1)                                                   SET02070
      ITEMP5=11                                                     SET02080
      IF(KR.EQ.1)GO TO 295                                          SET02090
      MN(4)=MN(8)                                                   SET02100
      MN(3)=MN(7)                                                   SET02110
      MN(1)=MN(5)                                                   SET02120
      MN(2)=MN(6)                                                   SET02130
  295 GO TO 500                                                     SET02140
C     FRONT TOP RIGHT CORNER (12)                                   SET02150
  300 IF(KR.EQ.KRM)GO TO 320                                        SET02160
      NPRP=NPR+4*IRM*JRM-2*IRM+1                                    SET02170
      ITEMP5=12                                                     SET02180
```

```
      HD(1)=HZ(KR+1)                                                    SET02190
      MN(6)=MMAP(IR,JR,KR+1)                                            SET02200
      MN(5)=MMAP(IR+1,JR,KR+1)                                          SET02210
      MN(8)=MMAP(IR+1,JR-1,KR+1)                                        SET02220
      MN(7)=MMAP(IR,JR-1,KR+1)                                          SET02230
      IF(JR.NE.1)GO TO 305                                              SET02240
      MN(8)=MN(5)                                                       SET02250
      MN(7)=MN(6)                                                       SET02260
  305 GO TO 500                                                         SET02270
C   TOP RIGHT EDGE (13)                                                 SET02280
  310 NPRP=NPR+4*IRM*JRM+1                                              SET02290
      ITEMP5=13                                                         SET02300
      IF(JR.EQ.1)GO TO 315                                              SET02310
      HD(4)=HD(3)                                                       SET02320
      MN(7)=MN(6)                                                       SET02330
      MN(8)=MN(5)                                                       SET02340
      MN(4)=MN(1)                                                       SET02350
      MN(3)=MN(2)                                                       SET02360
  315 GO TO 500                                                         SET02370
C   RIGHT SIDE (14)                                                     SET02380
  320 NPRP=NPR+1                                                        SET02390
      ITEMP5=14                                                         SET02400
      IF(KR.NE.KRM)GO TO 325                                            SET02410
      IF(JR.EQ.1)GO TO 325                                              SET02420
      HD(4)=HD(3)                                                       SET02430
      MN(4)=MN(1)                                                       SET02440
      MN(3)=MN(2)                                                       SET02450
  325 MN(8)=MN(4)                                                       SET02460
      MN(5)=MN(1)                                                       SET02470
      MN(7)=MN(3)                                                       SET02475
      MN(6)=MN(2)                                                       SET02480
      HD(1)=HD(2)                                                       SET02490
      GO TO 500                                                         SET02500
C   BACK BOTTOM RIGHT CORNER (15)                                       SET02510
  330 IF(JR.EQ.JRM)GO TO 420                                            SET02520
      NPRP=NPR-4*IRM*JRM+2*IRM+1                                        SET02530
```

```
      ITEMP5=15                                                         SET02540
      HD(3)=HY(JR+1)                                                    SET02550
      MN(5)=MMAP(IR+1,JR+1,KR)                                          SET02560
      MN(6)=MMAP(IR,JR+1,KR)                                            SET02570
      MN(2)=MN(6)                                                       SET02580
      MN(1)=MN(5)                                                       SET02590
      IF(KR.EQ.1)GO TO 335                                              SET02600
      HD(2)=HZ(KR-1)                                                    SET02610
      MN(2)=MMAP(IR,JR+1,KR-1)                                          SET02620
      MN(1)=MMAP(IR+1,JR+1,KR-1)                                        SET02630
      MN(3)=MMAP(IR,JR,KR-1)                                            SET02640
      MN(4)=MMAP(IR+1,JR,KR-1)                                          SET02650
  335 GO TO 500                                                         SET02660
C    BACK RIGHT EDGE (16)                                               SET02670
  340 NPRP=NPR+2*IRM+1                                                  SET02680
      ITEMP5=16                                                         SET02690
      IF(KR.EQ.1)GO TO 345                                              SET02700
      HD(2)=HZ(KR)                                                      SET02710
      MN(1)=MN(5)                                                       SET02720
      MN(2)=MN(6)                                                       SET02730
      MN(3)=MN(7)                                                       SET02740
      MN(4)=MN(8)                                                       SET02750
  345 GO TO 500                                                         SET02760
C    BACK TOP RIGHT CORNER (17)                                         SET02770
  350 IF(KR.EQ.KRM)GO TO 370                                            SET02775
      NPRP=NPR+4*IRM*JRM+2*IRM+1                                        SET02780
      ITEMP5=17                                                         SET02790
      HD(1)=HZ(KR+1)                                                    SET02800
      MN(5)=MMAP(IR+1,JR+1,KR+1)                                        SET02810
      MN(6)=MMAP(IR,JR+1,KR+1)                                          SET02820
      MN(7)=MMAP(IR,JR,KR+1)                                            SET02830
      MN(8)=MMAP(IR+1,JR,KR+1)                                          SET02840
      GO TO 500                                                         SET02850
C    BACK TOP EDGE (18)                                                 SET02860
  360 IF(JR.EQ.JRM)GO TO 420                                            SET02865
      IF(KR.EQ.KRM)GO TO 370                                            SET02870
```

```
      NPRP=NPR+4*IRM*JRM+2*IRM                                          SET02875
      ITEMP5=18                                                         SET02880
      IF(IR.NE.IRM)GO TO 365                                            SET02890
      HD(1)=HZ(KR+1)                                                    SET02900
      HD(2)=HZ(KR)                                                      SET02910
      HD(3)=HY(JR+1)                                                    SET02920
      MN(2)=MMAP(IR,JR+1,KR)                                            SET02930
      MN(3)=MMAP(IR,JR,KR)                                              SET02940
      MN(6)=MMAP(IR,JR+1,KR+1)                                          SET02950
      MN(7)=MMAP(IR,JR,KR+1)                                            SET02960
  365 MN(1)=MN(2)                                                       SET02970
      MN(4)=MN(3)                                                       SET02980
      MN(5)=MN(6)                                                       SET02990
      MN(8)=MN(7)                                                       SET03000
      HD(5)=HD(6)                                                       SET03010
      GO TO 500                                                         SET03020
C   BACK SIDE (19)                                                      SET03030
  370 NPRP=NPR+2*IRM                                                    SET03040
      ITEMP5=19                                                         SET03050
      IF(IR.NE.IRM.AND.KR.NE.KRM)GO TO 375                              SET03060
      MN(2)=MMAP(IR,JR+1,KR)                                            SET03062
      MN(3)=MMAP(IR,JR,KR)                                              SET03064
      HD(2)=HD(1)                                                       SET03066
      HD(3)=HY(JR+1)                                                    SET03068
      HD(5)=HD(6)                                                       SET03070
      MN(4)=MN(3)                                                       SET03080
      MN(1)=MN(2)                                                       SET03090
  375 HD(1)=HZ(KR)                                                      SET03100
      MN(5)=MN(1)                                                       SET03110
      MN(6)=MN(2)                                                       SET03120
      MN(7)=MN(3)                                                       SET03130
      MN(8)=MN(4)                                                       SET03140
      GO TO 500                                                         SET03150
C   BACK BOTTOM EDGE (20)                                               SET03160
  380 NPRP=NPR-4*IRM*JRM+2*IRM                                          SET03170
      ITEMP5=20                                                         SET03180
```

```
      IF(KR.EQ.1)GO TO 385
      HD(2)=HZ(KR-1)
      MN(1)=MMAP(IR,JR+1,KR-1)
      MN(2)=MN(1)
      MN(3)=MMAP(IR,JR,KR-1)
      MN(4)=MN(3)
  385 GO TO 500
C   BACK BOTTOM LEFT CORNER (21)
  390 NPRP=NPR-4*IRM*JRM+2*IRM-1
      ITEMP5=21
      IF(IR.EQ.1)GO TO 395
      HD(6)=HX(IR-1)
      MN(6)=MMAP(IR-1,JR+1,KR)
      MN(7)=MMAP(IR-1,JR,KR)
      MN(3)=MN(7)
      MN(2)=MN(6)
      IF(KR.EQ.1)GO TO 395
      MN(3)=MMAP(IR-1,JR,KR-1)
      MN(2)=MMAP(IR-1,JR+1,KR-1)
  395 GO TO 500
C   BACK LEFT EDGE (22)
  400 NPRP=NPR+2*IRM-1
      ITEMP5=22
      IF(KR.EQ.1)GO TO 405
      MN(1)=MN(5)
      MN(2)=MN(6)
      MN(3)=MN(7)
      MN(4)=MN(8)
      HD(2)=HD(1)
  405 GO TO 500
C   BACK TOP LEFT CORNER (23)
  410 IF(KR.EQ.KRM)GO TO 530
      NPRP=NPR+4*IRM*JRM+2*IRM-1
      ITEMP5=23
      HD(1)=HZ(KR+1)
      MN(5)=MMAP(IR,JR+1,KR+1)
```

```
      MN(8)=MMAP(IR,JR,KR+1)                                            SET03550
      MN(6)=MN(5)                                                       SET03560
      MN(7)=MN(8)                                                       SET03570
      IF(IR.EQ.1)GO TO 415                                              SET03580
      MN(6)=MMAP(IR-1,JR+1,KR+1)                                        SET03590
      MN(7)=MMAP(IR-1,JR,KR+1)                                          SET03600
  415 GO TO 500                                                         SET03610
C   TOP LEFT EDGE (24)                                                  SET03620
  420 IF(KR.EQ.KRM)GO TO 530                                            SET03630
      NPRP=NPR+4*IRM*JRM-1                                              SET03635
      ITEMP5=24                                                         SET03640
      IF(IR.NE.IRM.AND.JR.NE.JRM)GO TO 425                              SET03650
      HD(1)=HZ(KR+1)                                                    SET03660
      HD(2)=HZ(KR)                                                      SET03665
      HD(5)=HX(IR)                                                      SET03670
      MN(8)=MMAP(IR,JR,KR+1)                                            SET03680
      MN(4)=MMAP(IR,JR,KR)                                              SET03690
      MN(7)=MN(8)                                                       SET03700
      MN(3)=MN(4)                                                       SET03705
      IF(IR.EQ.1)GO TO 425                                              SET03710
      HD(6)=HX(IR-1)                                                    SET03720
      MN(7)=MMAP(IR-1,JR,KR+1)                                          SET03730
      MN(3)=MMAP(IR-1,JR,KR)                                            SET03740
  425 HD(3)=HD(4)                                                       SET03750
      MN(1)=MN(4)                                                       SET03760
      MN(2)=MN(3)                                                       SET03770
      MN(5)=MN(8)                                                       SET03780
      MN(6)=MN(7)                                                       SET03790
      GO TO 500                                                         SET03800
C   FRONT TOP LEFT CORNER                                               SET03810
  430 NPRP=NPR+4*IRM*JRM-2*IRM-1                                        SET03820
      ITEMP5=25                                                         SET03830
      IF(JR.EQ.1)GO TO 435                                              SET03840
      HD(4)=HY(JR-1)                                                    SET03850
      MN(4)=MMAP(IR,JR-1,KR)                                            SET03860
      MN(8)=MMAP(IR,JR-1,KR+1)                                          SET03870
```

```
      MN(3)=MN(4)
      MN(7)=MN(8)
      IF(IR.EQ.1)GO TO 435
      MN(3)=MMAP(IR-1,JR-1,KR)
      MN(7)=MMAP(IR-1,JR-1,KR+1)
  435 GO TO 500
C   TOP FRONT EDGE
  440 NPRP=NPR+4*IRM*JRM-2*IRM
      ITEMP5=26
      IF(IR.EQ.1)GO TO 445
      HD(6)=HD(5)
      MN(2)=MN(1)
      MN(3)=MN(4)
      MN(6)=MN(5)
      MN(7)=MN(8)
  445 GO TO 500
C   TOP SIDE (27)
  450 NPRP=NPR+4*IRM*JRM
      ITEMP5=27
      IF(JR.EQ.1)GO TO 455
      HD(4)=HD(3)
      MN(4)=MN(1)
      MN(3)=MN(2)
      MN(7)=MN(6)
      MN(8)=MN(5)
  455 GO TO 500
C   BRANCH HERE TO COMPUTE COEFFICIENTS
  500 NRP=NPRP
      TEMP3=1.0D0
      DO 520 NG=1,NNG
      DD1(NRP,NG)=((HD(3)*HD(2)/SIGT(MN(1),NG))+(HD(3)*HD(1)/SIGT(MN(5),
     1NG))+(HD(4)*HD(2)/SIGT(MN(4),NG))+(HD(4)*HD(1)/SIGT(MN(8),NG)))*
     2(TEMP3/(HD(5)*TEMP))
      DD2(NRP,NG)=((HD(5)*HD(2)/SIGT(MN(1),NG))+(HD(6)*HD(2)/SIGT(MN(2),
     1NG))+(HD(6)*HD(1)/SIGT(MN(6),NG))+(HD(5)*HD(1)/SIGT(MN(5),NG)))*
     2(TEMP3/(HD(3)*TEMP))
```

```
      DD3(NRP,NG)=((HD(4)*HD(6)/SIGT(MN(7),NG))+(HD(4)*HD(5)/SIGT(MN(8),SET04240
     1NG))+(HD(3)*HD(5)/SIGT(MN(5),NG))+(HD(3)*HD(6)/SIGT(MN(6),NG)))*  SET04250
     2(TEMP3/(HD(1)*TEMP))                                               SET04260
      DD4(NRP,NG)=DD1(NRP,NG)+DD2(NRP,NG)+DD3(NRP,NG)+(((HD(3)*HD(2)/SIGSET04270
     1T(MN(2),NG))+(HD(4)*HD(2)/SIGT(MN(3),NG))+(HD(4)*HD(1)/SIGT(MN(7),SET04280
     2NG))+(HD(3)*HD(1)/SIGT(MN(6),NG)))/(HD(6)*TEMP)+((HD(6)*HD(2)/SIGTSET04290
     3(MN(3),NG))+(HD(5)*HD(2)/SIGT(MN(4),NG))+(HD(5)*HD(1)/SIGT(MN(8),NSET04300
     4G))+(HD(6)*HD(1)/SIGT(MN(7),NG)))/(HD(4)*TEMP)+((HD(5)*HD(3)/SIGT(SET04310
     5MN(1),NG))+(HD(3)*HD(6)/SIGT(MN(2),NG))+(HD(4)*HD(6)/SIGT(MN(3),NGSET04320
     6))+(HD(4)*HD(5)/SIGT(MN(4),NG)))/(HD(2)*TEMP))*TEMP3               SET04330
      DD4(NRP,NG)=0.5D0*DD4(NRP,NG)                                      SET04340
      DD5(NRP,NG)=DD4(NRP,NG)+(HD(5)*HD(3)*HD(2)*SIGR(MN(1),NG)+HD(6)*HDSET04350
     1(3)*HD(2)*SIGR(MN(2),NG)+HD(6)*HD(4)*HD(2)*SIGR(MN(3),NG)+HD(5)*HDSET04360
     2(4)*HD(2)*SIGR(MN(4),NG)+HD(5)*HD(3)*HD(1)*SIGR(MN(5),NG)+HD(6)*HDSET04370
     3(3)*HD(1)*SIGR(MN(6),NG)+HD(6)*HD(4)*HD(1)*SIGR(MN(7),NG)+HD(5)*HDSET04380
     4(4)*HD(1)*SIGR(MN(8),NG))*(TEMP3/TEMP1)                            SET04390
      DD6(NRP,NG)=(HD(5)*HD(3)*HD(2)*SIGF(MN(1),NG)*XNU(MN(1),NG)+HD(6)*SET04400
     1HD(3)*HD(2)*SIGF(MN(2),NG)*XNU(MN(2),NG)+HD(6)*HD(4)*HD(2)*SIGF(MNSET04410
     2(3),NG)*XNU(MN(3),NG)+HD(5)*HD(4)*HD(2)*SIGF(MN(4),NG)*XNU(MN(4),NSET04420
     3G)+HD(5)*HD(3)*HD(1)*SIGF(MN(5),NG)*XNU(MN(5),NG)+HD(6)*HD(3)*HD(1SET04430
     4)*SIGF(MN(6),NG)*XNU(MN(6),NG)+HD(6)*HD(4)*HD(1)*SIGF(MN(7),NG)*XNSET04440
     5U(MN(7),NG)+HD(5)*HD(4)*HD(1)*SIGF(MN(8),NG)*XNU(MN(8),NG))*(TEMP3SET04450
     6/TEMP1)                                                            SET04460
      DD5(NRP,NG)=DD5(NRP,NG)-XIM(NG)*DD6(NRP,NG)                        SET04470
      IF(NG.EQ.NNG)GO TO 520                                             SET04471
      DO 510 NDN=1,NDNSCT                                                SET04480
      DD7(NRP,NG,NDN)=(HD(5)*HD(3)*HD(2)*SIGS(MN(1),NG,NDN)+HD(6)*HD(3)*SET04490
     1HD(2)*SIGS(MN(2),NG,NDN)+HD(6)*HD(4)*HD(2)*SIGS(MN(3),NG,NDN)+HD(5SET04500
     2)*HD(4)*HD(2)*SIGS(MN(4),NG,NDN)+HD(5)*HD(3)*HD(1)*SIGS(MN(5),NG,NSET04510
     3DN)+HD(5)*HD(3)*HD(1)*SIGS(MN(6),NG,NDN)+HD(6)*HD(4)*HD(1)*SIGS(MNSET04520
     4(7),NG,NDN)+HD(5)*HD(4)*HD(1)*SIGS(MN(8),NG,NDN))*(TEMP3/TEMP1)    SET04530
  510 CONTINUE                                                           SET04540
  520 CONTINUE                                                           SET04550
      GO TO (200,210,220,230,240,250,260,270,280,290,300,310,320,330,340SET04560
     1,350,360,370,380,390,400,410,420,430,440,450,530),ITEMP5           SET04570
  530 CONTINUE                                                           SET04580
```

```
  540 CONTINUE                                                      SET04590
  550 CONTINUE                                                      SET04600
  600 CONTINUE                                                      SET04610
      RETURN                                                        SET04620
      END                                                           SET04630
```

```
      SUBROUTINE DELAYS(ALAM,BETA,XIP,DD6,VO,NPRMP,PSI,P2,PSO,PO,NNGV,NDDEL00010
     1GV,NTOGV,NMATV,IMV,JMV,KMV,NPRGV)                                 DEL00020
      IMPLICIT REAL*8 (A-H,O-Z)                                         DEL00030
      INTEGER*2 MMAP,NPRMP                                              DEL00040
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   DEL00050
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    DEL00060
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,DEL00070
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     DEL00080
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    DEL00090
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    DEL00140
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           DEL00150
      COMMON/TIMINT/LASZON,ISTPCH,ILINCH,IPRSTP,MVSCH(5),MNLCH(5),      DEL00160
     1ISTEP,ICHHT                                                       DEL00170
      COMMON/TIMFLO/T,HT,HMIN,HMAX,TSTART,TEND,DELSFS(5,4),DELSRS(5,4), DEL00180
     1DELSTS(5,4),DELS1S(5,4),DELS2S(5,4),DELSFL(5,4),DELSRL(5,4),      DEL00190
     2DELSTL(5,4),DELS1L(5,4),DELS2L(5,4)                               DEL00200
      DIMENSION ALAM(NDGV),BETA(NDGV),XIP(NNGV,NDGV),DD6(NPRGV,NNGV),   DEL00210
     1NPRMP(IMV,JMV,KMV),PSI(NTOGV,IMV,JMV,KMV),P2(NTOGV,IMV,JMV),      DEL00220
     2PSO(IMV,JMV,KMV),PO(IMV,JMV),VO(NPRGV)                            DEL00230
      IF(IOPT.EQ.1)GO TO 200                                            DEL00240
      DO 180 K=1,KM                                                     DEL00250
      DO 110 NG=1,NNG                                                   DEL00260
      READ(IOSC1)((PSI(NG,I,J,K),I=1,IM),J=1,JM)                        DEL00270
      IF(NG.NE.NTG)GO TO 110                                            DEL00280
      DO 100 J=1,JM                                                     DEL00290
      DO 100 I=1,IM                                                     DEL00300
  100 PSO(I,J,K)=PSI(NTG,I,J,K)                                         DEL00310
  110 CONTINUE                                                          DEL00320
      NDL=NNG+1                                                         DEL00330
      DO 170 J=1,JM                                                     DEL00340
      DO 170 I=1,IM                                                     DEL00350
      IF(K.EQ.KM)GO TO 150                                              DEL00360
      NPR=NPRMP(I,J,K)                                                  DEL00370
      TEMP=0.0D0                                                        DEL00380
      DO 120 NG=1,NNG                                                   DEL00390
  120 TEMP=TEMP+DD6(NPR,NG)*PSI(NG,I,J,K)                               DEL00400
```

```
      TEMP=TEMP/(EFFK*VO(NPR))                                          DEL00405
      DO 140 NG=NDL,NTOG                                                DEL00410
      ND=NG-NNG                                                         DEL00420
      IF(J.EQ.JM.OR.I.EQ.IM)GO TO 130                                   DEL00430
      PSI(NG,I,J,K)=BETA(ND)*TEMP/ALAM(ND)                              DEL00440
      GO TO 140                                                         DEL00450
  130 PSI(NG,I,J,K)=0.0D0                                               DEL00460
  140 CONTINUE                                                          DEL00470
      GO TO 170                                                         DEL00480
  150 DO 160 NG=NDL,NTOG                                                DEL00490
  160 PSI(NG,I,J,KM)=0.0D0                                              DEL00500
  170 CONTINUE                                                          DEL00510
  180 CONTINUE                                                          DEL00520
      GO TO 300                                                         DEL00530
C   BRANCH HERE IF IOPT=1                                               DEL00540
  200 DO 280 K=1,KM                                                     DEL00550
      DO 210 NG=1,NNG                                                   DEL00560
      READ(IOSC1)((P2(NG,I,J),I=1,IM),J=1,JM)                           DEL00570
  210 CONTINUE                                                          DEL00580
      WRITE(IOFO)((P2(NTG,I,J),I=1,IM),J=1,JM)                          DEL00590
      NDL=NNG+1                                                         DEL00600
      DO 270 J=1,JM                                                     DEL00610
      DO 270 I=1,IM                                                     DEL00620
      IF(K.EQ.KM)GO TO 250                                              DEL00630
      NPR=NPRMP(I,J,K)                                                  DEL00640
      TEMP=0.0D0                                                        DEL00650
      DO 220 NG=1,NNG                                                   DEL00660
  220 TEMP=TEMP+DD6(NPR,NG)*P2(NG,I,J)/(EFFK*VO(NPR))                   DEL00670
      DO 240 NG=NDL,NTOG                                                DEL00680
      ND=NG-NNG                                                         DEL00690
      IF(I.EQ.IM.OR.J.EQ.JM)GO TO 230                                   DEL00700
      P2(NG,I,J)=BETA(ND)*TEMP/ALAM(ND)                                 DEL00710
      GO TO 240                                                         DEL00720
  230 P2(NG,I,J)=0.0D0                                                  DEL00730
  240 CONTINUE                                                          DEL00740
      GO TO 270                                                         DEL00750
```

```
250 DO 260 NG=NDL,NTOG
260 P2(NG,I,J)=0.0D0
270 CONTINUE
    WRITE(IOPO)P2
280 CONTINUE
    REWIND IOPO
    REWIND IOFO
300 REWIND IOSC1
    RETURN
    END
```

```
      SUBROUTINE STEPAO(V,XIM,ALAM,BETA,XIP,X,Y,Z,HX,HY,HZ,DD1,DD2,DD3, STE00010
     1DD4,DD5,DD6,DD7,VO,NPRMP,PSI,W,NNGV,NDGV,NTOGV,NDNSCV,IMV,JMV,KMV,STE00020
     2IRMV,JRMV,KRMV,NPRGV,NGXV)                                        STE00030
      IMPLICIT REAL*8 (A-H,O-Z)                                         STE00040
      INTEGER*2 MMAP,NPRMP                                              STE00050
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   STE00060
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    STE00070
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,STE00080
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     STE00090
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    STE00100
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    STE00150
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           STE00160
      COMMON/TIMINT/LASZON,ISTPCH,ILINCH,IPRSTP,MNSCH(5),MNLCH(5),      STE00170
     1ISTEP,ICHHT                                                       STE00180
      COMMON/TIMFLO/T,HT,HMIN,HMAX,TSTART,TEND,DELSFS(5,4),DELSRS(5,4), STE00190
     1DELSTS(5,4),DELS1S(5,4),DELS2S(5,4),DELSFL(5,4),DELSRL(5,4),      STE00200
     2DELSTL(5,4),DELS1L(5,4),DELS2L(5,4)                               STE00210
      DIMENSION V(NNGV),XIM(NNGV),ALAM(NDGV),BETA(NDGV),XIP(NNGV,NDGV), STE00220
     1X(IMV),Y(JMV),Z(KMV),HX(IRMV),HY(JRMV),HZ(KRMV),DD1(NPRGV,NNGV),  STE00230
     2DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),DD5(NPRGV,NNGV),  STE00240
     3DD6(NPRGV,NNGV),DD7(NPRGV,NGXV,NDNSCV),NPRMP(IMV,JMV,KMV),        STE00250
     4PSI(NTOGV,IMV,JMV,KMV),W(IMV,JMV,KMV),VO(NPRGV)                   STE00260
      DIMENSION CC(4,4),DD(4)                                           STE00270
C   FIRST TRANSFORM ALL POINTS                                          STE00280
      DO 110 K=2,KMX                                                    STE00290
      DO 110 J=1,JMX                                                    STE00300
      DO 110 I=1,IMX                                                    STE00310
      TEMP1=DEXP(W(I,J,K)*HT)                                           STE00320
      DO 100 NG=1,NNG                                                   STE00330
  100 PSI(NG,I,J,K)=TEMP1*PSI(NG,I,J,K)                                 STE00340
  110 CONTINUE                                                          STE00350
C   NOW SET STARTING I,J,AND K INDICES ASSUMING NO SYMMETRY BOUNDARIES  STE00360
      IS=2                                                              STE00370
      JS=2                                                              STE00380
      KS=2                                                              STE00390
      HINV=1.0D0/HT                                                     STE00400
```

```
500 DO 850 K=KS,KMX                                                  STE02370
    IF(NFBC.EQ.0)GO TO 660                                           STE02380
    DO 620 NG=1,NNG                                                  STE02390
    TEMP=1.0D0/(V(NG)*HT)                                            STE02400
    DO 550 J=1,2                                                     STE02410
    DO 550 I=1,2                                                     STE02420
    NPR=NPRMP(I,J,K)                                                 STE02430
    II=2*(J-1)+I                                                     STE02440
    DD(II)=0.0D0                                                     STE02450
    DO 520 NGP=1,NTOG                                                STE02460
    IF(NGP.GT.NNG)GO TO 510                                          STE02470
    IF(NGP.EQ.NG)GO TO 520                                           STE02480
    DD(II)=DD(II)+XIM(NG)*DD6(NPR,NGP)*PSI(NGP,I,J,K)                STE02490
    GO TO 520                                                        STE02500
510 ND=NGP-NNG                                                       STE02510
    DD(II)=DD(II)+XIP(NG,ND)*PSI(NGP,I,J,K)*VO(NPR)                  STE02520
520 CONTINUE                                                         STE02530
    DO 530 NDN=1,NDNSCT                                              STE02540
    ITEMP1=NG-NDN                                                    STE02550
    IF(ITEMP1.LE.0)GO TO 530                                         STE02560
    DD(II)=DD(II)+DD7(NPR,ITEMP1,NDN)*PSI(ITEMP1,I,J,K)              STE02570
530 CONTINUE                                                         STE02580
    PTEM=TEMP*VO(NPR)                                                STE02581
    DD(II)=DD(II)+(PTEM-DD4(NPR,NG))*PSI(NG,I,J,K)+DD1(NPR,NG)*PSI(NG,STE02590
   1I+1,J,K)+DD2(NPR,NG)*PSI(NG,I,J+1,K)+DD3(NPR,NG)*PSI(NG,I,J,K+1)+ STE02600
   2DD3(NPRMP(I,J,K-1),NG)*PSI(NG,I,J,K-1)                           STE02610
    DO 540 ITEMP2=1,4                                                STE02620
540 CC(II,ITEMP2)=0.0D0                                              STE02630
    CC(II,II)=(TEMP+W(I,J,K)/V(NG))*VO(NPR)+DD5(NPR,NG)              STE02640
550 CONTINUE                                                         STE02650
    NPR=NPRMP(1,1,K)                                                 STE02660
    CC(1,2)=-DD1(NPR,NG)                                             STE02670
    CC(1,3)=-DD2(NPR,NG)                                             STE02680
    CC(2,4)=-DD2(NPRMP(2,1,K),NG)                                    STE02690
    CC(3,4)=-DD1(NPRMP(1,2,K),NG)                                    STE02700
    CC(2,1)=CC(1,2)                                                  STE02710
```

```
      CC(3,1)=CC(1,3)                                                   STE02720
      CC(4,2)=CC(2,4)                                                   STE02730
      CC(4,3)=CC(3,4)                                                   STE02740
C     CALL DSIMQ TO SOLVE SYSTEM                                        STE02750
      NEQ=4                                                             STE02760
      CALL DSIMQ(CC,DD,NEQ,ISING)                                       STE02770
      DO 560 J=1,2                                                      STE02780
      DO 560 I=1,2                                                      STE02790
      II=2*(J-1)+I                                                      STE02800
  560 PSI(NG,I,J,K)=DD(II)                                              STE02810
      DO 620 I=3,IMX                                                    STE02820
      DO 570 II=1,4                                                     STE02830
  570 DD(II)=0.0D0                                                      STE02840
      NPRY=NPRMP(I,1,K)                                                 STE02850
      DO 610 J=1,2                                                      STE02860
      NPR=NPRMP(I,J,K)                                                  STE02870
      DO 590 NGP=1,NTOG                                                 STE02880
      IF(NGP.GT.NNG)GO TO 580                                           STE02890
      IF(NGP.EQ.NG)GO TO 590                                            STE02900
      DD(J)=DD(J)+XIM(NG)*DD6(NPR,NGP)*PSI(NGP,I,J,K)                   STE02910
      GO TO 590                                                         STE02920
  580 ND=NGP-NNG                                                        STE02930
      DD(J)=DD(J)+XIP(NG,ND)*PSI(NGP,I,J,K)*VO(NPR)                     STE02940
  590 CONTINUE                                                          STE02950
      DO 600 NDN=1,NDNSCT                                               STE02960
      ITEMP1=NG-NDN                                                     STE02970
      IF(ITEMP1.LE.0)GO TO 600                                          STE02980
      DD(J)=DD(J)+DD7(NPR,ITEMP1,NDN)*PSI(ITEMP1,I,J,K)                 STE02990
  600 CONTINUE                                                          STE03000
      PTEM=TEMP*VO(NPR)                                                 STE03001
      DD(J)=DD(J)+(PTEM-DD4(NPR,NG))*PSI(NG,I,J,K)+DD1(NPR,NG)*PSI(NG,I+STE03010
     1I,J,K)+DD2(NPR,NG)*PSI(NG,I,J+1,K)+DD1(NPRMP(I-1,J,K),NG)*PSI(NG,ISTE03020
     2-1,J,K)+DD3(NPR,NG)*PSI(NG,I,J,K+1)+DD3(NPRMP(I,J,K-1),NG)*PSI(NG,STE03030
     3I,J,K-1)                                                          STE03040
  610 DD(J+2)=(TEMP+W(I,J,K)/V(NG))*VO(NPR)+DD5(NPR,NG)                 STE03051
      TEMP5=DD(3)*DD(4)-(DD2(NPRY,NG)**2.0D0)                           STE03060
```

```
      PSI(NG,I,1,K)=(DD(1)*DD(4)+DD(2)*DD2(NPRY,NG))/TEMP5             STE03070
      PSI(NG,I,2,K)=(DD(2)*DD(3)+DD(1)*DD2(NPRY,NG))/TEMP5             STE03080
  620 CONTINUE                                                          STE03090
      DO 650 J=1,2                                                      STE03100
      DO 650 I=1,IMX                                                    STE03110
      NPR=NPRMP(I,J,K)                                                  STE03120
      DO 640 ND=1,NDG                                                   STE03130
      NG=ND+NNG                                                         STE03140
      TEMP1=0.0D0                                                       STE03150
      DO 630 NGP=1,NNG                                                  STE03160
  630 TEMP1=TEMP1+BETA(ND)*DD6(NPR,NGP)*PSI(NGP,I,J,K)                  STE03170
      TEMP1=TEMP1/(EFFK*VO(NPR))                                        STE03180
  640 PSI(NG,I,J,K)=((HINV-ALAM(ND))*PSI(NG,I,J,K)+TEMP1)/(HINV+ALAM(ND)STE03190
     1)                                                                 STE03200
  650 CONTINUE                                                          STE03210
      JS=3                                                              STE03220
  660 DO 840 J=JS,JMX                                                   STE03230
      IF(NLBC.EQ.0)GO TO 760                                            STE03240
      DO 720 NG=1,NNG                                                   STE03250
      TEMP=1.0D0/(V(NG)*HT)                                             STE03260
      DO 670 II=1,4                                                     STE03270
  670 DD(II)=0.0D0                                                      STE03280
      NPRX=NPRMP(1,J,K)                                                 STE03290
      DO 710 I=1,2                                                      STE03300
      NPR=NPRMP(I,J,K)                                                  STE03310
      DO 690 NGP=1,NTOG                                                 STE03320
      IF(NGP.GT.NNG)GO TO 680                                           STE03330
      IF(NGP.EQ.NG)GO TO 690                                            STE03340
      DD(I)=DD(I)+XIM(NG)*DD6(NPR,NGP)*PSI(NGP,I,J,K)                   STE03350
      GO TO 690                                                         STE03360
  680 ND=NGP-NNG                                                        STE03370
      DD(I)=DD(I)+XIP(NG,ND)*PSI(NGP,I,J,K)*VO(NPR)                     STE03380
  690 CONTINUE                                                          STE03390
      DO 700 NDN=1,NDNSCT                                               STE03400
      ITEMP1=NG-NDN                                                     STE03410
      IF(ITEMP1.LE.0)GO TO 700                                          STE03420
```

```
      DD(I)=DD(I)+DD7(NPR,ITEMP1,NDN)*PSI(ITEMP1,I,J,K)                 STE03430
  700 CONTINUE                                                          STE03440
      PTEM=TEMP*VO(NPR)                                                 STE03441
      DD(I)=DD(I)+(PTEM-DD4(NPR,NG))*PSI(NG,I,J,K)+DD1(NPR,NG)*PSI(NG,I+STE03450
     11,J,K)+DD2(NPR,NG)*PSI(NG,I,J+1,K)+DD2(NPRMP(I,J-1,K),NG)*PSI(NG,ISTE03460
     2,J-1,K)+DD3(NPR,NG)*PSI(NG,I,J,K+1)+DD3(NPRMP(I,J,K-1),NG)*PSI(NG STE03470
     3,I,J,K-1)                                                         STE03480
  710 DD(I+2)=(TEMP+W(I,J,K)/V(NG))*VO(NPR)+DD5(NPR,NG)                 STE03490
      TEMP5=DD(3)*DD(4)-(DD1(NPRX,NG)**2.0D0)                           STE03500
      PSI(NG,1,J,K)=(DD(1)*DD(4)+DD(2)*DD1(NPRX,NG))/TEMP5              STE03510
      PSI(NG,2,J,K)=(DD(2)*DD(3)+DD(1)*DD1(NPRX,NG))/TEMP5              STE03520
  720 CONTINUE                                                          STE03530
      DO 750 I=1,2                                                      STE03540
      NPR=NPRMP(I,J,K)                                                  STE03550
      DO 740 ND=1,NDG                                                   STE03560
      NG=ND+NNG                                                         STE03570
      TEMP1=0.0D0                                                       STE03580
      DO 730 NGP=1,NNG                                                  STE03590
  730 TEMP1=TEMP1+BETA(ND)*DD6(NPR,NGP)*PSI(NGP,I,J,K)                  STE03600
      TEMP1=TEMP1/(EFFK*VO(NPR))                                        STE03610
  740 PSI(NG,I,J,K)=((HINV-ALAM(ND))*PSI(NG,I,J,K)+TEMP1)/(HINV+ALAM(ND)STE03620
     1)                                                                 STE03630
  750 CONTINUE                                                          STE03640
      IS=3                                                              STE03650
  760 DO 830 I=IS,IMX                                                   STE03660
      NPR=NPRMP(I,J,K)                                                  STE03670
      NPRX=NPRMP(I-1,J,K)                                               STE03680
      NPRY=NPRMP(I,J-1,K)                                               STE03690
      NPRZ=NPRMP(I,J,K-1)                                               STE03700
      DO 800 NG=1,NNG                                                   STE03710
      TEMP=1.0D0/(V(NG)*HT)                                             STE03720
      TEMP1=0.0D0                                                       STE03730
      DO 780 NGP=1,NTOG                                                 STE03740
      IF(NGP.GT.NNG)GO TO 770                                           STE03750
      IF(NGP.EQ.NG)GO TO 780                                            STE03760
      TEMP1=TEMP1+XIM(NG)*DD6(NPR,NGP)*PSI(NGP,I,J,K)                   STE03770
```

```
      GO TO 780                                                         STE03780
  770 ND=NGP-NNG                                                        STE03790
      TEMP1=TEMP1+XIP(NG,ND)*PSI(NGP,I,J,K)*VO(NPR)                     STE03800
  780 CONTINUE                                                          STE03810
      DO 790 NDN=1,NDNSCT                                               STE03820
      ITEMP1=NG-NDN                                                     STE03830
      IF(ITEMP1.LE.0)GO TO 790                                          STE03840
      TEMP1=TEMP1+DD7(NPR,ITEMP1,NDN)*PSI(ITEMP1,I,J,K)                 STE03850
  790 CONTINUE                                                          STE03860
      PTEM=TEMP*VO(NPR)                                                 STE03861
      PSI(NG,I,J,K)=((PTEM-DD4(NPR,NG))*PSI(NG,I,J,K)+DD1(NPR,NG)*PSI(NGSTE03870
     1,I+1,J,K)+DD1(NPRX,NG)*PSI(NG,I-1,J,K)+DD2(NPR,NG)*PSI(NG,I,J+1,K)STE03880
     2+DD2(NPRY,NG)*PSI(NG,I,J-1,K)+DD3(NPR,NG)*PSI(NG,I,J,K+1)+DD3(NPRZSTE03890
     3,NG)*PSI(NG,I,J,K-1)+TEMP1)/((TEMP+W(I,J,K)/V(NG))*VO(NPR)+DD5(NPRSTE03900
     4,NG))                                                             STE03901
  800 CONTINUE                                                          STE03910
      DO 820 ND=1,NDG                                                   STE03920
      NG=ND+NNG                                                         STE03930
      TEMP1=0.0D0                                                       STE03940
      DO 810 NGP=1,NNG                                                  STE03950
  810 TEMP1=TEMP1+BETA(ND)*DD6(NPR,NGP)*PSI(NGP,I,J,K)                  STE03960
      TEMP1=TEMP1/(EFFK*VO(NPR))                                        STE03970
  820 PSI(NG,I,J,K)=((HINV-ALAM(ND))*PSI(NG,I,J,K)+TEMP1)/(HINV+ALAM(ND)STE03980
     1)                                                                 STE03990
  830 CONTINUE                                                          STE04000
  840 CONTINUE                                                          STE04010
  850 CONTINUE                                                          STE04020
      RETURN                                                            STE04030
      END                                                               STE04040
```

```
      SUBROUTINE STEPBO(V,XIM,ALAM,BETA,XIP,X,Y,Z,HX,HY,HZ,DD1,DD2,DD3, STE00010
     1DD4,DD5,DD6,DD7,VO,NPRMP,PSI,W,NNGV,NDGV,NTOGV,NDNSCV,IMV,JMV,KMV,STE00020
     2IRMV,JRMV,KRMV,NPRGV,NGXV)                                        STE00030
      IMPLICIT REAL*8 (A-H,O-Z)                                         STE00040
      INTEGER*2 MMAP,NPRMP                                              STE00050
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,   STE00060
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),    STE00070
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,STE00080
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,     STE00090
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX                    STE00100
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,    STE00150
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT           STE00160
      COMMON/TIMINT/LASZON,ISTPCH,ILINCH,IPRSTP,MNSCH(5),MNLCH(5),      STE00170
     1ISTEP,ICHHT                                                       STE00180
      COMMON/TIMFLO/T,HT,HMIN,HMAX,TSTART,TEND,DELSFS(5,4),DELSRS(5,4), STE00190
     1DELSTS(5,4),DELS1S(5,4),DELS2S(5,4),DELSFL(5,4),DELSRL(5,4),      STE00200
     2DELSTL(5,4),DELS1L(5,4),DELS2L(5,4)                               STE00210
      DIMENSION V(NNGV),XIM(NNGV),ALAM(NDGV),BETA(NDGV),XIP(NNGV,NDGV), STE00220
     1X(IMV),Y(JMV),Z(KMV),HX(IRMV),HY(JRMV),HZ(KRMV),DD1(NPRGV,NNGV),  STE00230
     2DD2(NPRGV,NNGV),DD3(NPRGV,NNGV),DD4(NPRGV,NNGV),DD5(NPRGV,NNGV),  STE00240
     3DD6(NPRGV,NNGV),DD7(NPRGV,NGXV,NDNSCV),NPRMP(IMV,JMV,KMV),        STE00250
     4PSI(NTOGV,IMV,JMV,KMV),W(IMV,JMV,KMV),VO(NPRGV)                   STE00260
      KE=KMX-1                                                          STE00270
      JE=JMX-1                                                          STE00290
      IF(NFBC.EQ.1)JE=JMX-2                                             STE00300
      HINV=1.0D0/HT                                                     STE00310
      IE=IMX-1                                                          STE00320
      DO 340 KK=1,KE                                                    STE00330
      K=KM-KK                                                           STE00340
      DO 220 JJ=1,JE                                                    STE00350
      J=JM-JJ                                                           STE00360
      DO 210 II=1,IE                                                    STE00370
      I=IM-II                                                           STE00380
      NPR=NPRMP(I,J,K)                                                  STE00390
      NPRX=NPRMP(I-1,J,K)                                               STE00400
      NPRY=NPRMP(I,J-1,K)                                               STE00410
```

```
      NPRZ=NPRMP(I,J,K-1)                                                STE00420
      DO 160 NG=1,NNG                                                    STE00430
      TEMP=1.0D0/(V(NG)*HT)                                              STE00440
  110 TEMP1=0.0D0                                                        STE00450
      DO 130 NGP=1,NTOG                                                  STE00460
      IF(NGP.GT.NNG)GO TO 120                                            STE00470
      IF(NGP.EQ.NG)GO TO 130                                             STE00480
      TEMP1=TEMP1+XIM(NG)*DD6(NPR,NGP)*PSI(NGP,I,J,K)                    STE00490
      GO TO 130                                                          STE00500
  120 ND=NGP-NNG                                                         STE00510
      TEMP1=TEMP1+XIP(NG,ND)*PSI(NGP,I,J,K)*VO(NPR)                      STE00520
  130 CONTINUE                                                           STE00530
      DO 140 NDN=1,NDNSCT                                                STE00540
      ITEMP1=NG-NDN                                                      STE00550
      IF(ITEMP1.LE.0)GO TO 140                                           STE00560
      TEMP1=TEMP1+DD7(NPR,ITEMP1,NDN)*PSI(ITEMP1,I,J,K)                  STE00570
  140 CONTINUE                                                           STE00580
      PTEM=TEMP*VO(NPR)                                                  STE00581
      IF(I.EQ.1)GO TO 150                                                STE00590
      TEMP2=PSI(NG,I,J,K)                                                STE00600
      PSI(NG,I,J,K)=((PTEM-DD4(NPR,NG))*TEMP2+DD1(NPR,NG)*PSI(NG,I+1,J,KSTE00610
     1)+DD1(NPRX,NG)*PSI(NG,I-1,J,K)+DD2(NPR,NG)*PSI(NG,I,J+1,K)+DD2(NPRSTE00620
     2Y,NG)*PSI(NG,I,J-1,K)+DD3(NPR,NG)*PSI(NG,I,J,K+1)+DD3(NPRZ,NG)*PSISTE00630
     3(NG,I,J,K-1)+TEMP1)/((TEMP+W(I,J,K)/V(NG))*VO(NPR)+DD5(NPR,NG))   STE00640
      IF(I.GT.2)GO TO 160                                                STE00650
      IF(NLBC.EQ.0)GO TO 160                                             STE00660
      I=1                                                                STE00670
      NPR=NPRMP(1,J,K)                                                   STE00680
      GO TO 110                                                          STE00690
  150 PSI(NG,1,J,K)=((PTEM-DD4(NPR,NG))*PSI(NG,1,J,K)+DD1(NPR,NG)*(PSI(NSTE00700
     1G,2,J,K)+TEMP2)+DD2(NPR,NG)*PSI(NG,1,J+1,K)+DD2(NPRMP(1,J-1,K),NG)STE00710
     2*PSI(NG,1,J-1,K)+DD3(NPR,NG)*PSI(NG,1,J,K+1)+DD3(NPRMP(1,J,K-1),NGSTE00720
     3)*PSI(NG,1,J,K-1)+TEMP1)/((TEMP+W(I,J,K)/V(NG))*VO(NPR)+DD5(NPR,NGSTE00730
     4))                                                                 STE00731
      NPR=NPRMP(2,J,K)                                                   STE00740
      I=2                                                                STE00750
```

```
160 CONTINUE
    DO 200 ND=1,NDG
    NG=ND+NNG
170 TEMP1=0.0D0
    DO 180 NGP=1,NNG
180 TEMP1=TEMP1+BETA(ND)*DD6(NPR,NGP)*PSI(NGP,I,J,K)
    TEMP1=TEMP1/(EFFK*VO(NPR))
    PSI(NG,I,J,K)=((HINV-ALAM(ND))*PSI(NG,I,J,K)+TEMP1)/(HINV+ALAM(ND)
   1)
    IF(I.EQ.1)GO TO 190
    IF(I.GT.2)GO TO 200
    IF(NLBC.EQ.0)GO TO 200
    I=1
    NPR=NPRMP(1,J,K)
    GO TO 170
190 I=2
    NPR=NPRMP(2,J,K)
200 CONTINUE
210 CONTINUE
220 CONTINUE
    IF(NFBC.EQ.0)GO TO 340
    DO 300 NG=1,NNG
    TEMP=1.0D0/(V(NG)*HT)
    DO 290 II=1,IE
230 TEMP2=PSI(NG,I,1,K)
    DO 270 JJ=1,2
    J=3-JJ
    NPR=NPRMP(I,J,K)
    NPRX=NPRMP(I-1,J,K)
    NPRY=NPRMP(I,1,K)
    NPRZ=NPRMP(I,J,K-1)
    TEMP1=0.0D0
    DO 250 NGP=1,NTOG
    IF(NGP.GT.NNG)GO TO 240
    IF(NGP.NE.NG)TEMP1=TEMP1+DD6(NPR,NGP)*PSI(NGP,I,J,K)
    IF(NGP.EQ.NNG)TEMP1=TEMP1*XIM(NG)
```

```
      GO TO 250                                                         STE01120
  240 ND=NGP-NNG                                                        STE01130
      TEMP1=TEMP1+XIP(NG,ND)*PSI(NGP,I,J,K)*VO(NPR)                     STE01140
  250 CONTINUE                                                          STE01150
      TEMP3=PSI(NG,I,2,K)                                               STE01160
      PTEM=TEMP*VO(NPR)                                                 STE01161
      IF(I.EQ.1)GO TO 260                                               STE01170
      PSI(NG,I,J,K)=((PTEM-DD4(NPR,NG))*PSI(NG,I,J,K)+DD1(NPR,NG)*PSI(NGSTE01180
     1,I+1,J,K)+DD1(NPRX,NG)*PSI(NG,I-1,J,K)+DD2(NPR,NG)*PSI(NG,I,J+1,K)STE01190
     2+DD2(NPRY,NG)*TEMP2+DD3(NPR,NG)*PSI(NG,I,J,K+1)+DD3(NPRZ,NG)*PSI(NSTE01200
     3G,I,J,K-1)+TEMP1)/((TEMP+W(I,J,K)/V(NG))*VO(NPR)+DD5(NPR,NG))     STE01210
      TEMP2=TEMP3                                                       STE01220
      GO TO 270                                                         STE01230
  260 PSI(NG,I,J,K)=((PTEM-DD4(NPR,NG))*PSI(NG,I,J,K)+DD1(NPR,NG)*(TEMP4STE01240
     1+PSI(NG,2,J,K))+DD2(NPR,NG)*PSI(NG,1,J+1,K)+DD2(NPRY,NG)*TEMP2+   STE01250
     2DD3(NPR,NG)*PSI(NG,1,J,K+1)+DD3(NPRZ,NG)*PSI(NG,1,J,K-1)+TEMP1)/  STE01260
     3((TEMP+W(1,J,K)/V(NG))*VO(NPR)+DD5(NPR,NG))                       STE01270
      TEMP4=TEMP5                                                       STE01280
      TEMP2=TEMP3                                                       STE01290
  270 CONTINUE                                                          STE01300
      IF(I.EQ.2)GO TO 280                                               STE01310
      IF(I.NE.3)GO TO 290                                               STE01320
      TEMP4=PSI(NG,2,2,K)                                               STE01330
      TEMP5=PSI(NG,2,1,K)                                               STE01340
      GO TO 290                                                         STE01350
  280 I=1                                                               STE01360
      GO TO 230                                                         STE01370
  290 CONTINUE                                                          STE01380
  300 CONTINUE                                                          STE01390
      DO 330 II=1,IMX                                                   STE01400
      I=IM-II                                                           STE01410
      DO 330 JJ=1,2                                                     STE01420
      J=3-JJ                                                            STE01430
      NPR=NPRMP(I,J,K)                                                  STE01440
      DO 320 ND=1,NDG                                                   STE01450
      NG=ND+NNG                                                         STE01460
```

```
      TEMP1=0.0D0                                                       STE01470
      DO 310 NGP=1,NNG                                                  STE01480
  310 TEMP1=TEMP1+BETA(ND)*DD6(NPR,NGP)*PSI(NGP,I,J,K)                  STE01490
      TEMP1=TEMP1/(EFFK*VO(NPR))                                        STE01500
  320 PSI(NG,I,J,K)=((HINV-ALAM(ND))*PSI(NG,I,J,K)+TEMP1)/(HINV+ALAM(ND)STE01510
     1)                                                                 STE01520
  330 CONTINUE                                                          STE01530
  340 CONTINUE                                                          STE01540
C   NOW CARRY OUT EXP(W*H) TRANSFORMATION                               STE01550
  350 DO 370 K=2,KMX                                                    STE01560
      DO 370 J=1,JMX                                                    STE01570
      DO 370 I=1,IMX                                                    STE01580
      TEMP1=DEXP(W(I,J,K)*HT)                                           STE01590
      DO 360 NG=1,NNG                                                   STE01600
  360 PSI(NG,I,J,K)=TEMP1*PSI(NG,I,J,K)                                 STE01610
  370 CONTINUE                                                          STE01620
      RETURN                                                            STE01630
      END                                                               STE01640
```

```
      SUBROUTINE FREQO(PSI,PSO,W,NTOGV,IMV,JMV,KMV)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER*2 MMAP,NPRMP
      COMMON/INTG/IASIZE,NNG,NDG,NTOG,NMAT,IM,JM,KM,IRM,JRM,KRM,NLBC,
     1NFBC,NBBC,NDNSCT,NPRG,IOPT,NTG,NXTP,NYTP,NZTP,IXTP(5),IYTP(5),
     2IZTP(5),NSTEAD,IFLIN,IGEOM,ITITLE(20),NOIT,NIIT,NPIT,IOPSI,IODUMP,
     3IOFN,IOFO,IOPN,IOPO,ITEMP,ITEMP1,ITEMP2,ITEMP3,ITEMP4,ITEMP5,
     4NTIT,IETIME,IFLOUT,IMX,JMX,KMX,IOSC1,IOSC2,NGX
      COMMON/FLOTE/EFFK,ORFP,EPS1,EPS2,TEMP,TEMP1,TEMP2,TEMP3,TEMP4,
     1TEMP5,TEMP6,XFISST,XFISSO,ALAMN,ALAMO,TIME,FLXCON,BETAT
      COMMON/TIMINT/LASZON,ISTPCH,ILINCH,IPRSTP,MNSCH(5),MNLCH(5),
     1ISTEP,ICHHT
      COMMON/TIMFLO/T,HT,HMIN,HMAX,TSTART,TEND,DELSFS(5,4),DELSRS(5,4),
     1DELSTS(5,4),DELS1S(5,4),DELS2S(5,4),DELSFL(5,4),DELSRL(5,4),
     2DELSTL(5,4),DELS1L(5,4),DELS2L(5,4)
      DIMENSION PSI(NTOGV,IMV,JMV,KMV),PSO(IMV,JMV,KMV),W(IMV,JMV,KMV)
      TEMP5=1.0D0/(2.0D0*HT)
C   COMPUTE FREQUENCIES
      DO 120 K=2,KMX
      DO 120 J=1,JMX
      DO 120 I=1,IMX
      IF(PSO(I,J,K).LT.1.0D-30)GO TO 110
      TEMP4=PSI(NTG,I,J,K)/PSO(I,J,K)
      IF(DABS(1.0D0-TEMP4).LT.1.0D-08)GO TO 110
      W(I,J,K)=TEMP5*DLOG(TEMP4)
      GO TO 120
  110 W(I,J,K)=0.0D0
  120 CONTINUE
      RETURN
      END
```