VARIATIONAL DERIVATION OF MODAL-NODAL FINITE DIFFERENCE EQUATIONS IN SPATIAL REACTOR PHYSICS

by

PATRICK G. BAILEY

B.S., University of California at Berkeley
(1967)

M.S., Massachusetts Institute of Technology
(1969)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF DOCTOR OF
PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF
TECHNOLOGY

July, 1972

Signature of Author ....
Department of Nuclear Engineering, July 18, 1972

Certified by ....
Thesis Supervisor

Certified by ....
Thesis Supervisor

Accepted by ....
Chairman, Departmental Committee
on Graduate Students

VARIATIONAL DERIVATION OF MODAL-NODAL FINITE DIFFERENCE EQUATIONS IN SPATIAL REACTOR PHYSICS

by

PATRICK G. BAILEY

Submitted to the Department of Nuclear Engineering on July 18, 1972, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ABSTRACT

A class of consistent coarse mesh modal-nodal approximation methods is presented for the solution of the spatial neutron flux in multigroup diffusion theory. The methods are consistent in that they are systematically derived as an extension of the finite element method by utilizing general modal-nodal variational techniques. Detailed subassembly solutions, found by imposing zero current boundary conditions over the surface of each subassembly, are modified by piecewise continuous Hermite polynomials of the finite element method and used directly in trial function forms. Methods using both linear and cubic Hermite basis functions are presented and discussed.

The proposed methods differ substantially from the finite element methods in which homogeneous nuclear constants, homogenized by flux weighting with detailed subassembly solutions, are used. However, both schemes become equivalent when the subassemblies themselves are homogeneous.

One-dimensional, two-group numerical calculations using representative PWR nuclear material constants and 18-cm subassemblies were performed using entire subassemblies as coarse mesh regions. The results indicate that the proposed methods can yield comparable if not superior criticality measurements, comparable regional power levels, and extremely accurate subassembly fine flux structure with little increase of computational effort in comparison with existing coarse mesh methods.

Thesis Supervisors: Allan F. Henry
Kent F. Hansen

Titles: Professors of Nuclear Engineering

ACKNOWLEDGMENTS

I would like to express my deep appreciation and sincerest gratitude to Dr. Allan F. Henry whose advice and consultation have been of considerable help during the course of this work. Thanks must also be extended to Dr. Kent F. Hansen, who reviewed the final manuscript and provided many beneficial discussions and programming assistance.

It has also been a privilege to have known and worked with the late Dr. T. J. Thompson, whose untimely death was a great loss to the nation as well as to the department at M.I.T., and Dr. Elias P. Gyftopoulos. Their insistence on high standards of education and quality as well as clarity of expression will continue to remain with me always.

Fellowship support by the U.S. Atomic Energy Commission is also gratefully acknowledged. The work was performed under USAEC Contract AT (30-1) - 3903. All computations were performed on IBM 360/65 and 370/155 computers at the M.I.T. Information Processing Center.

Special appreciation is extended to my wife, Alice, whose patience and understanding have provided immeasurable support throughout the course of this work.

Finally, special thanks are expressed to Mrs. Mary Bosco, who has once again demonstrated her expert ability as a superb technical typist.

. . . Until one is committed there is hesitancy, the chance to draw back, always ineffectiveness. Concerning all acts of initiative (and creation), there is one elementary truth, the ignorance of which kills countless ideas and splendid plans: that the moment one definitely commits oneself, then Providence moves too. All sorts of things occur to help one that would never otherwise have occurred. A whole stream of events issues from the decision, raising in one's favor all manner of unforeseen incidents and meetings and material assistance, which no man could have dreamt would have come his way. I have learned a deep respect for one of Goethe's couplets:

> Whatever you can do, or dream you can, begin it.
> Boldness has genius, power, and magic in it.

— W.H. Murray

Great importance attaches to the material comforts of life, and equanimity, unconcern, security are all sacrificed to them. The American lives even more for his goals, for the future, than the European. Life for him is always becoming, never being.

— Albert Einstein 1921

"When someone is seeking," said Siddhartha, "it happens quite easily that he only sees the thing that he is seeking; that he is unable to absorb anything, because he is only thinking of the thing he is seeking, because he has a goal, because he is obsessed with his goal. Seeking means: to have a goal; but finding means: to be free, to be receptive, to have no goal. You, O worthy one, are perhaps indeed a seeker, for in striving toward your goal, you do not see many things that are under your nose."

— Hermann Hesse

TABLE OF CONTENTS

LIST OF FIGURES

No.

No.

LIST OF TABLES

Chapter 1
INTRODUCTION

1.1 Preface

The large variety of approximation methods and techniques used in computational reactor analysis and simulation has caused the area of numerical reactor physics to become one of the most exciting areas in applied nuclear reactor physics today. The application of numerical analysis is most important in two phases of reactor design; feasibility studies and safety analysis. The primary consideration of the reactor physicist has been and must continue to be the safety of the reactor during and after any foreseeable nuclear accident. A realistic safety analysis can be obtained only if all the physical processes occuring within the reactor can be adequately described and related. Since all of these processes can be shown to be dependent upon the neutron density distribution throughout the reactor core, a detailed solution of the spatial neutron flux is vital.[1]

The dynamic characteristics of a reactor strongly depend upon the spatial approximation and solution of the neutron flux. Approximation methods utilizing gross averaging of the flux near localized strong absorption and production regions, such as cruciform control rods or small water channels, can lead to inaccurate results. Large errors may result from the use of such methods in spatial kinetics problems such as depletion and xenon oscillation calculations. Much attention has therefore been focused upon approximation methods which can obtain detailed spatial neutron flux distributions within large reactor cores.

The Boltzmann neutron transport equation[2] is considered to be a sufficiently detailed description of the physical processes occuring within a nuclear reactor, and naturally is most difficult to solve. The P-1 and diffusion theory approximations[3] greatly simplify the transport equation into more tractable equations which have been found to approximate adequately the flux distributions for most large-core reactors such as PWR, BWR, and LMFBR core geometries. The advent of high speed digital computers has enabled widespread use of diffusion theory because of its simple mathematical form and straightforward numerical solution techniques inherent with its use.

The treatment of the spatial approximation in diffusion theory is the primary concern of this report. There is in existence an increasingly abundant variety of such approximation methods currently in use. Fine mesh methods,[4] for example, can yield very accurate results through the use of extremely large numbers of unknowns. However, such methods may well exceed the storage capacity of present day computers, as well as being exceedingly costly. Coarse mesh methods and particularly synthesis techniques,[5] on the other hand, have recently become attractive as the number of unknowns can be drastically reduced, although the accuracy of many of these methods is in doubt.

The purpose of this report is twofold: first, to present the general development of variational approximation methods used to derive difference approximations to the neutron diffusion equation; and second, to extend this development in order to develop systematically a class of consistent coarse mesh approximation methods which can approximate accurately the detailed spatial neutron flux and can also be easily incorporated into present day computer codes. As this report will deal only with the spatial approximation, the inclusion of time dependence will be set aside for future study.

1.2 The Time-Independent, Multigroup Diffusion Theory Equations

The energy discretized multigroup P-1 approximation to the Boltzmann neutron transport equation excluding time dependence can be written in standard group notation for each energy group g as follows:[3]

$$\underline{j}_g(r) + D_g(r)\,\underline{\nabla}\phi_g(r) = 0 \tag{1.1a}$$

$$\underline{\nabla}\cdot\underline{j}_g(r) + \Sigma_g(r)\phi_g(r) - \sum_{\substack{g'=1\\ g'\neq g}}^{G} \Sigma_{gg'}(r)\phi_{g'}(r) = \frac{1}{\lambda}\chi_g \sum_{g'=1}^{G} \nu\Sigma_{fg'}(r)\phi_{g'}(r) \tag{1.1b}$$

where the group index g runs from the highest energy group, 1, to the lowest energy group, G. The symbols and notation used throughout this report are summarized in Appendix A. Equations 1.1 are the standard P-1 equations which relate the vector neutron current $\underline{j}_g(r)$ for each energy group g with the scalar neutron flux $\phi_g(r)$. The current may be eliminated via Fick's law, Eq. 1.1a, in order to obtain the multigroup diffusion equation:

$$-\underline{\nabla}\cdot D_g(r)\underline{\nabla}\phi_g(r) + \Sigma_g(r)\phi_g(r) - \sum_{\substack{g'=1\\ g'\neq G}}^{G} \Sigma_{gg'}(r)\phi_{g'}(r) = \frac{1}{\lambda}\chi_g \sum_{g'=1}^{G} \nu\Sigma_{fg'}(r)\phi_{g'}(r) \tag{1.2}$$

Equation 1.2 can be written in operator matrix notation as

$$-\underline{\nabla} \cdot \mathbb{D}(r)\underline{\nabla}\Phi(r) + [\mathbb{M}(r) - \mathbb{T}(r)]\Phi(r) = \frac{1}{\lambda}\mathbb{B}(r)\Phi(r) \tag{1.3}$$

where $\mathbb{D}$, $\mathbb{M}$, $\mathbb{T}$, and $\mathbb{B}$ are $G \times G$ group matrices defined by

$$\mathbb{D}(r) = \mathrm{Diag}[D_1(r) \ldots D_g(r) \ldots D_G(r)] \tag{1.4a}$$

$$\mathbb{M}(r) = \mathrm{Diag}[\Sigma_1(r) \ldots \Sigma_g(r) \ldots \Sigma_G(r)] \tag{1.4b}$$

$$\mathbb{T}(r) = \begin{bmatrix} 0 & -\Sigma_{12}(r) & \cdots & -\Sigma_{1G}(r) \\ -\Sigma_{21}(r) & 0 & \cdots & -\Sigma_{2G}(r) \\ \vdots & \vdots & & \vdots \\ -\Sigma_{G1}(r) & -\Sigma_{G2}(r) & \cdots & 0 \end{bmatrix} \tag{1.4c}$$

$$\mathbb{B}(r) = \begin{pmatrix} \chi_1 \\ \vdots \\ \chi_G \end{pmatrix} \left[\nu\Sigma_{f1}(r) \ldots \nu\Sigma_{fG}(r) \right] \tag{1.4d}$$

and $\Phi(r)$ is the group flux vector

$$\Phi(r) = \mathrm{Col}[\phi_1(r) \ldots \phi_G(r)] \tag{1.4e}$$

In problems where no upscattering is present, $\Sigma_{gg'}(r) = 0$ for $g < g'$, and $\mathbb{T}$ becomes $G \times G$ lower triangular.

It is also convenient to define the group current vector $\underline{J}(r)$

$$\underline{J}(r) = \mathrm{Col}[\underline{j}_1(r) \ldots \underline{j}_G(r)] \tag{1.4f}$$

and the $G \times G$ group absorption, scattering and production matrix $\mathbf{\Lambda}(r)$

$$\mathbf{\Lambda}(r) = \mathbb{M}(r) - \mathbb{T}(r) - \frac{1}{\lambda}\mathbb{B}(r) \tag{1.4g}$$

Equations 1.1 and 1.2 may then be written simply as

$$\underline{J}(r) + \mathbb{D}(r)\,\underline{\nabla}\Phi(r) = 0 \tag{1.5a}$$

$$\underline{\nabla} \cdot \underline{J}(r) + \mathbf{\Lambda}(r)\,\Phi(r) = 0 \tag{1.5b}$$

and

$$-\underline{\nabla} \cdot \mathbb{D}(r)\,\underline{\nabla}\Phi(r) + \mathbf{\Lambda}(r)\,\Phi(r) = 0 \tag{1.6}$$

respectively. These forms of the group diffusion equations will be used throughout this report. The boundary conditions on $\Phi(r)$ are of the homogeneous Neumann or Dirichlet type,[6] while the normal component of the current $\underline{J}(r)$ is required to be continuous across all internal interfaces.

1.3 Solution Methods

All of the solution methods which can be employed in order to obtain approximate solutions to the time-dependent, multigroup diffusion equations may be conveniently classified as belonging in the area of either nodal analysis or modal analysis, or a combination of the two: modal-nodal analysis. The principal concept in each of these analyses is that the neutron flux, a continuous function of many variables, may be approximated as a set of unknown coefficients and/or functions of possibly fewer variables. The ultimate goals of such approximation methods are to produce easily solvable coupled equations which relate the unknowns to each approximation and yield results of acceptable accuracy at a low cost. Various commonly used methods and their drawbacks are discussed below.

1.3.1 Nodal Methods

Nodal methods involve the local approximation of an average flux at points called nodes, where each node represents a distinct region within the reactor in which the average flux is defined. An ordered set of nodes connected by a grid of mesh lines is then used to approximate the spatial flux behavior. The accuracy of such methods is generally governed by the internodal coupling or neutron current approximation inherent in each method.

A. Conventional Finite Difference Equations

The common finite difference equations used in diffusion theory can be derived using Taylor series expansion, variational techniques, or box integration methods about each spatial node.[7] The second-order diffusion term at each node is replaced by three-point difference equations relating consecutive nodes in each spatial direction. The resulting band-structured matrix equations exhibit many advantageous mathematical properties and can be solved with the use of simple solution algorithms.

The attractiveness of these difference equations is further enhanced by the fact that, for properly posed problems (including proper boundary conditions), the approximation can be shown to converge to the solution of the differential equation as the mesh size approaches zero. Also, the accuracy of the approximation can be shown to be in general of order $\theta(h)$,[8] thus error estimates for the approximation are available. It is for these reasons that these equations are frequently invoked as "exact" solutions to diffusion equation problems. The main disadvantage, however, is that, as the number of nodes increases, the amount of labor and cost involved in order to obtain an accurate solution increases geometrically. A point of diminishing returns is then quickly reached where further accuracy is prohibitively expensive. Another disadvantage is that any known physical insight or *a priori* detailed flux behavior cannot be used with this approximation.

A formal derivation of the conventional difference equations is given in section 2.3 of Chapter 2.

B. Gross Coupling Models

In gross coupling or coarse mesh nodal techniques an attempt is made to decrease drastically the number of nodes needed for solution without significantly decreasing solution accuracy. Many such methods have been proposed by postulating various forms of neutronic coupling or communication interaction between nodes.

1. Phenomenological Model[9,10,11]

From a physical viewpoint, the reactor can be divided into several distinct regions, each represented by a node located somewhere in that region. Equations of balance relating state variables of interest (average neutron flux, regional power, etc.) can then be written for each region and between region nodes. Internodal coupling is governed by a set of coefficients, say p_{ij}, which may account for the number of neutrons born in region i which appear in region j. A set of algebraic equations can then be written which describe the coupled core dynamics of the nodal interactions.

The principal drawback of such methods lies in the definition of the interaction parameters p_{ij}. Although the describing equations of the phenomenological model can be directly formulated from diffusion theory,[12] the method of calculating the coefficients p_{ij} remains unclear. However, the physical simplicity of this model has made it very appealing in coupled kinetics methods development. Much of the work in this field is based on deriving approximations which reduce to this simple conceptual model.

2. Effective D/L Coupling[13]

These methods are very similar to finite difference approximations in that the structural forms of the resulting difference equations are identical. In order to compensate for the use of large internodal mesh spacing, the reactor constants, and the diffusion coefficients in particular, may be altered so that they correspond in an average sense to those obtained from a fine mesh calculation.[14] In this way it is hoped that the gross internodal coupling will be sufficiently improved to compensate for the large mesh spacings.

It has been shown that such methods can indeed improve internodal coupling for large mesh regions; however, the results are generally not satisfactory since the coupling constants are dependent in an unpredictable way on changes in the properties of the nodes.

3. Fission Source Coupling[15]

The assumption that the reactor flux can be separated into partial region fluxes due to nodal fission sources permits a consistent derivation of nodal coupled kinetics equations from multigroup diffusion theory. Fission modes can be found from detailed flux solutions which are then used to account for internodal coupling. This method gives reasonably accurate results for fast and thermal reactor transients, although the number of nodes necessary to achieve an accurate solution must increase as the form of the spatial flux becomes more detailed.

4. Multichannel Coupling[16]

By partitioning the reactor into regions called channels and allowing only adjacent channel-to-channel interactions, coupling coefficients p_{ij} can be found which represent the net leakage of neutrons from channel i into channel j in terms of the corresponding averaged channel fluxes. The coupling coefficients can be calculated using diffusion theory or variational techniques which yield the diffusion equations as stationary conditions. This model is appealing in that it can be shown to reduce to the conventional difference equations when a regular grid of small channel regions is used.

The above examples of gross coupling models are generally unsatisfactory because they require the use of average fluxes defined within large regions of the reactor. More acceptable results are obtained by utilizing known or *a priori* detailed spatial flux shapes in the regions in the approximation method.

1.3.2 Modal Methods[17]

Modal methods imply an extensive rather than local approximation to the spatial neutron flux. In general the flux is represented by a combination of known functions defined over the regions of interest with unknown functions as mixing coefficients. Depending upon the approximation employed, relationships among these coefficients can be derived which are hopefully simpler to solve than the original equation.

A. Helmholtz Modes[18]

The diffusion equation for a completely homogeneous reactor formally has an infinite solution set of eigenvalues and corresponding orthogonal eigenfunctions, called Helmholtz modes, which satisfy the homogeneous boundary conditions. For the general case of a heterogeneous reactor, the spatially dependent flux can be approximated as a linear combination of these modes. The major difficulty with this approach is that a large number of modes is required in order to approximate the solution flux, and thus the appeal for this simplistic modal approach is quickly lost.

B. Lambda[19] and Omega Modes[20]

Although included in the class of modal approximations, these methods require the use of known spatial solutions for time-dependent analysis. Lambda modes belong to the set of detailed flux solutions of the time-independent diffusion equations which correspond to different lambda eigenvalues.

A set of detailed flux solutions can also be found from the time-dependent diffusion equations by allowing the time-dependent flux to be separable and given in the form $e^{\omega t}$. The solutions of the resulting equations, called ω modes, correspond to different omega eigenvalues.

Both of these methods have successfully been used in the transient analysis of coupled nodal kinetics.

C. Synthesis Methods[5,21]

The use of synthesis techniques for the derivation of modal approximations is the most exciting and fastest growing area of reactor analysis methods development. This can be attributed to the fact that all diffusion theory approximation schemes, both modal and nodal, and those including time dependence, can be ultimately derived from one single variational principle. Each approximation scheme is therefore dependent solely upon the form of the trial functions used to represent the flux, current, and weighting functions (or adjoint functions) in the synthesis procedure. The outstanding advantage of the synthesis method is that knowledge of *a priori* detailed flux shapes or other physical insights can be incorporated directly into the approximation method.

1. Multichannel Synthesis[22]

This method may be viewed as a modal extension of the multichannel gross coupling method. Assuming the flux to be separable in its variables (x,y,z), the number of unknowns can be reduced by specifying detailed flux shapes in any dimension. A common example assumes that in each channel, k, of the reactor the flux trial function, $U_k(x,y,z)$, can be expressed as the product of a known transverse flux, $\psi_k(x,y)$, with an unknown spatially dependent axial flux, $\rho_k(z)$, as:

$$U_k(x,y,z) = \rho_k(z)\,\psi_k(x,y) \tag{1.7}$$

The specification of the flux in two dimensions reduces the problem to an approximation involving only one dimension. If, however, the flux is approximated by a full spatial solution times an unknown constant,

$$U_k(x,y,z) = F_k \psi_k(x,y,z) \tag{1.8}$$

the method reduces to an approximation similar to the multichannel nodal method. Figure 1.1 illustrates the resulting flux shape characteristics of such a method for the one-dimensional case.

The major disadvantage of these multichannel synthesis methods lies in the fact that in general the flux is discontinuous at channel interfaces.[23,24] Therefore the adjacent channel coupling currents, which then must be continuous across these interfaces,† are defined in terms of averaged channel fluxes. Although these methods can produce detailed flux distributions in each channel, their accuracy appears to be not much better than nodal multichannel methods because of the averaged gross neutronic coupling requirements inherent in these methods.[25]

2. Overlapping Multichannel Synthesis[26,27]

The interchannel neutronic coupling can be improved by requiring that the flux trial functions be continuous across channel interfaces. This can be accomplished by modulating the known expansion functions, ψ_k, by piecewise continuous normalized polynomial functions, p_k, which are nonzero only within coupled channels

†Variational techniques used with diffusion theory in general do not allow the flux and current to be simultaneously discontinuous. Further clarification is given in section 2.2 of Chapter 2.

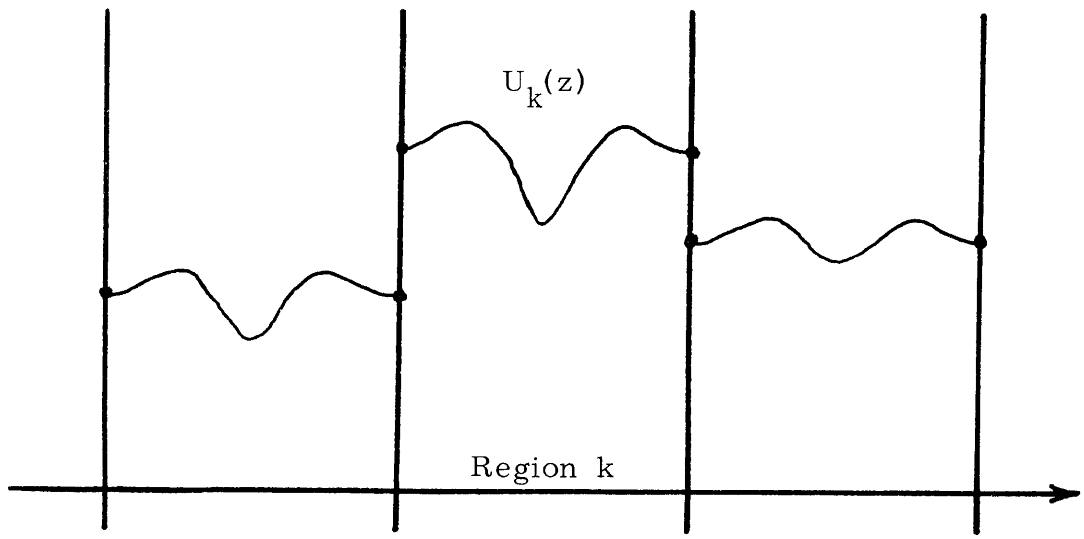


Figure 1.1. Illustration of One-Dimensional Multichannel Synthesis

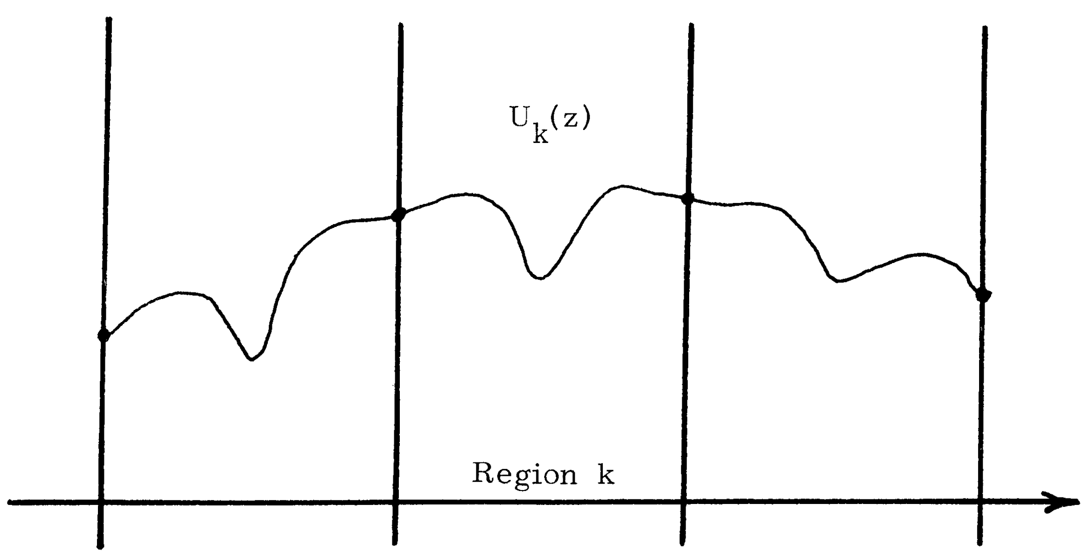


Figure 1.2. Illustration of One-Dimensional Overlapping Multichannel Synthesis

of interest, providing the expansion functions are continuous over all channels for which the corresponding polynomial functions are nonzero. Such polynomials are required to be normalized to unity at the coupling interface and zero along the external boundary of the channels in order to preserve flux trial function continuity.

In one dimension represented by the continuous variable z and K mesh regions bounded by the nodes z_k where $k = 1$ to $K+1$, for example, the simple linear functions

$$p_k(z) = \begin{cases} \dfrac{z - z_{k-1}}{z_k - z_{k-1}} & z_{k-1} \leqslant z \leqslant z_k \\[2ex] \dfrac{z_{k+1} - z}{z_{k+1} - z_k} & z_k \leqslant z \leqslant z_{k+1} \\[2ex] 0 & \text{otherwise} \end{cases} \tag{1.9}$$

satisfy these conditions. The flux can then be approximated as

$$U(z) = \sum_{k=1}^{K} F_k \, p_k(z) \psi_k(z) \tag{1.10}$$

where the set of F_k's are the unknowns of the method. The resulting flux shape characteristics of this approximation are illustrated in Figure 1.2.

Approximations based on this synthesis method are dependent upon the class of overlapping polynomial functions used as well as the form of the current trial functions employed. The form of the current is extremely important in that it specifies the coupling interaction between regions and in this sense governs the usefulness and accuracy of the approximation. Work performed with this method to date has used current trial functions of a form similar to those of the flux trial function. Although the results of these investigations have been encouraging, such methods do not reduce to more simple known approximation methods. In addition, the band-structured matrix equations which arise from the use of such methods do not exhibit mathematical properties desired of such approximation schemes and may be difficult and costly to solve.

1.3.3 Modal-Nodal Methods

Approximation methods have also been developed in which the flux has a known extensive definition, or shape, and the unknowns are local flux values averaged in accordance with their corresponding extensive definition. Such modal-nodal methods retain all of the advantages of modal methods while generally reducing the number of unknowns and producing matrix equations which have desirable mathematical properties for numerical approximation and solution.

The finite element method is the best example of a modal-nodal approximation. Greater accuracy than that of conventional difference techniques can be obtained by allowing the flux in each region of interest to be represented as a polynomial which is continuous at region interfaces. The forms of the flux approximations and the resulting difference equations which arise from the use of the finite element method are described in detail in section 2.3 of Chapter 2.

The purpose of this report is to present an original and consistent class of modal-nodal coarse mesh approximation methods which retain given or known detailed flux structure within the regions of interest, while providing detailed neutronic coupling between adjacent regions. These methods are consistent in that they are derived from a general variational principle and are a systematic extension of the finite element method as applied to diffusion theory reactor analysis.

For purposes of simplicity, the methods will be developed for the case of one-dimensional, time-dependent, multigroup diffusion theory, although it is expected that these methods can be extended to the general spatially dependent kinetics problem with relative ease.

The remainder of this report is organized as follows. Chapter 2 summarizes the use of variational principles and synthesis techniques in time-independent diffusion theory. The difference equations of the finite element methods applied in one dimension are derived using modal-nodal trial function forms in order to illustrate the use of these techniques. The forms of the proposed approximation methods are given in Chapter 3. The resulting finite difference equations are presented and boundary conditions discussed for approximation methods involving both linear and cubic Hermite basis functions. The numerical properties of the resulting matrix equations, as well as their numerical solution scheme, and useful programming techniques are discussed in Chapter 4. Chapter 5 presents results of the proposed methods for four representative one-dimensional PWR configurations, and compares the results with those of coarse mesh finite element methods. Finally, Chapter 6 presents conclusions and recommendations as well as comments concerning the possibility of extending the proposed methods to multidimensional geometries.

Chapter 2

VARIATIONAL DERIVATION OF FINITE DIFFERENCE APPROXIMATIONS IN TIME-INDEPENDENT MULTIGROUP DIFFUSION THEORY

The application of variational calculus to the describing equations of physical systems is perhaps the most general and powerful method of obtaining approximate solutions in mathematical physics. Variational methods seek to combine known "trial functions" into approximate solutions through the use of a variational functional which characterizes the equations of the system.

Essentially, variational methods consist of first finding a characteristic functional whose first-order variation when set to zero yields the describing equations of the system as its Euler equations. A class of trial functions, given in terms of known functions and unknown coefficients (or functions), is then chosen to approximate the solutions of the describing equations. These trial functions are then substituted into the variational functional, and its first variation is set to zero. Allowing arbitrary variations in all of the trial function unknowns results in a set of relationships among the unknowns. These relationships when solved then yield the "best" obtainable approximate solution within the space of trial functions given.

Variational methods can be thought of as a class of weighted residual methods since "weighting functions" appear in the functional and in the equations that result from setting the first variation of the functional to zero. The weighting functions are determined by the form of the functional itself; or equivalently, by the set of Euler equations selected to describe the system. In non-self adjoint problems, the adjoint equations are generally included in the set of Euler equations. The inclusion of corresponding "adjoint trial functions" in the functional results in adjoint weighting in the variation equations and allows greater approximation flexibility of the variational method.

2.1 Calculus of Variations Applied to Diffusion Theory

The time-independent multigroup diffusion equations as given by Eq. 1.3 can be written as

$$\mathbb{H}\Phi = \frac{1}{\lambda}\mathbb{B}\Phi \tag{2.1a}$$

where

$$\mathbb{H} = -\underline{\nabla}\cdot\mathbb{D}\underline{\nabla} + \mathbb{M} - \mathbf{T} \tag{2.1b}$$

Since the multigroup diffusion equations are not self-adjoint, it is convenient to introduce the adjoint diffusion equations

$$\mathbb{H}^*\Phi^* = \frac{1}{\lambda}\mathbb{B}^*\Phi^* \tag{2.2a}$$

where $\mathbb{H}^*$ and $\mathbb{B}^*$ are the adjoint operators corresponding to $\mathbb{H}$ and $\mathbb{B}$, respectively, and are defined as:[3]

$$\mathbb{H}^* = \mathbb{H}^T = -\underline{\nabla}\cdot\mathbb{D}\,\underline{\nabla} + \mathbb{M} - \mathbf{T}^T \tag{2.2b}$$

$$\mathbb{B}^* = \mathbb{B}^T \tag{2.2c}$$

since $\mathbb{D}$ and $\mathbb{M}$ are diagonal. Φ^* is the group adjoint flux vector, or importance vector, which must obey the same boundary conditions as Φ.[28]

The exact solutions $\Phi(r)$ and $\Phi^*(r)$ of the diffusion equations and the adjoint diffusion equations can be approximated by flux and adjoint flux trial functions denoted as U(r) and $U^*(r)$ using a variational functional of the form

$$\mathscr{F}_1 [U, U^*] = \frac{1}{\lambda} = \frac{\int_R U^{*T} \mathbb{H} U \, dr}{\int_R U^{*T} \mathbb{B} U \, dr} \tag{2.3}$$

where it is assumed that the group-theory flux trial function vectors U^* and U as well as the group current vectors $\mathbb{D}\underline{\nabla}U^*$ and $\mathbb{D}\underline{\nabla}U$ are everywhere continuous, and that U^* and U vanish outside the reactor region R. Allowing arbitrary trial function variations, denoted by δU^* and δU, making $\mathscr{F}_1$ stationary first with respect to U^* and then with respect to U results[29] in the following equations:

$$\int_R \delta U^{*T} [\mathbb{H} U - \frac{1}{\lambda} \mathbb{B} U] \, dr = 0 \tag{2.4a}$$

$$\int_R [U^{*T} \mathbb{H} - \frac{1}{\lambda} U^{*T} \mathbb{B}] \, \delta U \, dr = 0 \tag{2.4b}$$

The above equations, containing the desired Euler equations, are the equations upon which the approximation method is based.

A significant characteristic of this approximation form is the property of exact solution reproduction. Although general choices of the trial functions U and U^* result in approximate eigenvalues which may differ substantially from the exact solution eigenvalue, the exact solutions, when chosen within the given class of trial functions, are yielded as the result of the approximation along with the exact solution eigenvalue.

The nature of the above approximation depends solely upon the forms of the flux trial functions given. Each trial function can be defined in terms of unknown coefficients (or functions) and known functions. Independent variation of the unknown coefficients of the adjoint trial function in Eq. 2.4a will yield the "best" flux solution obtainable for that class of flux and adjoint flux trial functions given. The corresponding "best" adjoint flux solution can be found in an analogous manner using Eq. 2.4b. These techniques are illustrated in the next section.

Another functional incorporating the flux and adjoint flux diffusion equations can be defined as

$$\mathcal{F}_2[U^*, U] = \int_R U^{*T} [\mathbb{H}U - \frac{1}{\lambda}\mathbb{B}U]\, dr \qquad (2.5)$$

Although the forms of the above functionals differ, it can be shown that both produce the same variation equations, Eqs. 2.4, when made stationary. The form of $\mathcal{F}_2$ and its first variation are much less complex than the form and first variation of $\mathcal{F}_1$. For these reasons, functionals of the form of $\mathcal{F}_2$ will be used in this report.

2.2 Discontinuous Trial Functions

The addition of discontinuous flux trial functions into the class of allowable trial functions for use in diffusion theory variational methods greatly enhances and generalizes the versatility of such methods.[25] However, special provisions must be made in the approximation method itself in order that such trial functions can be properly used.[23,24,30] In order to account for the discontinuities in the flux (and in general also the current) trial functions, it is necessary to include special terms specifying continuity conditions directly within the approximation method. This can be accomplished through the use of a variational functional whose Euler equations include the P-1 equations and continuity conditions for both flux and current. A general functional of this type which allows discontinuous flux, current, and adjoint trial functions can be derived from previous work[30,31] and is given as follows:

$$\mathscr{F}[U^*, U, \underline{V}^*, \underline{V}, \alpha, \beta] = \int_R \{U^{*^T}[\underline{\nabla}\cdot\underline{V} + \mathbb{A}U] + \underline{V}^{*^T}\cdot[\underline{\nabla}U + \mathbb{D}^{-1}\underline{V}]\}dr$$

$$+ \int_\Gamma \hat{n}\cdot\{[U_+^{*^T}\alpha + U_-^{*^T}(I-\alpha)](\underline{V}_+ - \underline{V}_-)$$

$$+ [\underline{V}_+^{*^T}\beta + \underline{V}_-^{*^T}(I-\beta)](U_+ - U_-)\}ds \qquad (2.6)$$

where U^*, U, $\underline{V}^*$, and $\underline{V}$ are the group flux and group current approximations to Φ^*, Φ, $\underline{J}^*$, and $\underline{J}$, respectively, and where the first integral extends over the volume R of the reactor and the second extends over all interior surfaces Γ upon which discontinuities are defined. $\hat{n}$ is the unit vector perpendicular to interior surfaces, and quantities evaluated on sides of surfaces toward which $\hat{n}$ is pointing are denoted with the subscript (+). Quantities evaluated on sides of surfaces from which $\hat{n}$ is pointing are denoted with the subscript (-). α and β are in general $G \times G$ undefined variable matrices, and I is in general a $G \times G$ unit matrix, which allow a general treatment of the discontinuities.

The restrictions generally imposed upon trial functions for use in functionals of this type are the following:

1. The trial functions must be piecewise continuous.

2. The trial functions U and $\underline{V}^*$ as well as U^* and $\underline{V}$ are not allowed to be discontinuous at the same point.

3. The components of $U^T\underline{V}^*$ and $U^{*T}\underline{V}$ normal to the exterior surface of the reactor must vanish.

Due to restriction 2, the general quantities α and β always cancel and are never used within these approximation methods.

The first variation of $\mathcal{F}$ can be found in a straightforward manner, and can be simplified to the following form which indicates the desired P-1 and adjoint P-1 equations and the trial function continuity conditions as Euler equations:

$$\begin{aligned}\delta\mathcal{F} = \int_R \{ & \delta U^{*T}[\underline{\nabla}\cdot\underline{V}+\boldsymbol{\Lambda}U] + \delta\underline{V}^{*T}\cdot[\underline{\nabla}U+\mathbb{D}^{-1}\underline{V}] \\ & + [-\underline{\nabla}U^{*T}+\underline{V}^{*T}\mathbb{D}^{-1}]\cdot\delta\underline{V} + [-\underline{\nabla}\cdot\underline{V}^{*T}+U^{*T}\boldsymbol{\Lambda}]\delta U\}dr \\ & + \int_\Gamma \hat{n}\cdot\{\delta U^{*T}(\underline{V}_+-\underline{V}_-)+\delta\underline{V}^{*T}(U_+-U_-) \\ & \qquad + (U_-^*-U_+^*)^T\delta\underline{V}+(\underline{V}_-^*-\underline{V}_+^*)\delta U\}ds\end{aligned} \tag{2.7}$$

In most applications, only approximations to the flux and current solutions are desired. In such instances variations in only the adjoint trial functions need be taken. Setting the first variation of $\mathcal{F}$ equal to zero under these conditions and imposing the above trial function restrictions results in the following variation equation for flux and current approximations:

$$\int_R \{\delta U^{*T} [\underline{\nabla} \cdot \underline{V} + \Lambda U] + \delta \underline{V}^{*T} \cdot [\underline{\nabla} U + \mathbb{D}^{-1} \underline{V}]\} dr$$

$$+ \int_\Gamma \hat{n} \cdot \{\delta U^{*T} (\underline{V}_+ - \underline{V}_-) + \delta \underline{V}^{*T} (U_+ - U_-)\} ds = 0 \tag{2.8}$$

The above approximation can also be expressed independently of adjoint trial functions. If the adjoint trial functions are defined as

$$U^* = U \tag{2.9a}$$

$$\underline{V}^* = -\underline{V} \tag{2.9b}$$

then Eq. 2.8 reduces to the Rayleigh-Ritz Galerkin method, a weighted residual method based upon flux weighting.

Regardless of the choice of weighting, the variation equations can be further simplified for those approximation methods which require the currents to obey explicitly Fick's laws:

$$\underline{V} = -\mathbb{D} \underline{\nabla} U \tag{2.10a}$$

$$\underline{V}^* = +\mathbb{D} \underline{\nabla} U^* \tag{2.10b}$$

Under these conditions the variation equations for discontinuous flux and discontinuous current trial functions reduce to

$$\int_R \{\delta U^{*T} \Lambda U - \delta \underline{V}^{*T} \cdot \mathbb{D}^{-1} \underline{V}\} dr$$

$$+ \int_\Gamma \hat{n} \cdot \{(\delta U_-^* - \delta U_+^*)^T \underline{V} + \delta \underline{V}^{*T} (U_+ - U_-)\} ds = 0 \tag{2.11}$$

If in addition the flux is required to be everywhere continuous, the variation equations reduce to the appealing forms

$$\int_R \{\delta U^{*T} \Lambda U - \delta \underline{V}^{*T} \cdot \mathbb{D}^{-1} \underline{V}\} dr = 0 \tag{2.12a}$$

or equivalently

$$\int_R \{\delta U^{*^T} \Lambda U + (\underline{\nabla}\,\delta U^*)^T \cdot \mathbb{D}(\underline{\nabla}\, U)\} dr = 0 \qquad (2.12b)$$

Variation equations 2.11 and 2.12 are the approximation equations which are used with the finite element methods and the proposed approximation methods.

2.3 The Finite Element Approximation Methods[32,33]

This section introduces the notation and techniques used in conjunction with the modal-nodal variational analysis of the finite element method approximations in one-dimensional multigroup diffusion theory. These fundamentals are presented in these simple approximations before applying them to the more general proposed approximation method in the next chapter.

The one-dimensional problem is defined by the continuous variable z and divided into K adjoining regions which are in general inhomogeneous. Each region k is bounded by nodes z_k and z_{k+1} and has width $h_k = z_{k+1} - z_k$. It is convenient to define the dimensionless variable x within each region k as

$$x = \frac{z - z_k}{h_k} \qquad (2.13a)$$

so that region k can be described in terms of z as

$$z_k \leq z \leq z_k + h_k = z_{k+1} \qquad (2.13b)$$

or equivalently in terms of x as

$$0 \leq x \leq 1 \qquad (2.13c)$$

for each of the regions k, $k = 1$ to K. This notation will be used throughout this report.

2.3.1 The Conventional Finite Difference Equations

The conventional nodal flux-averaged, three-point, finite difference equations of one-dimensional diffusion theory can be derived from Eq. 2.8 using discontinuous flux and current multigroup column vector trial functions of the following form:[7,8 †]

$$\left.\begin{array}{l} U(z) = F_k \\ \\ U^*(z) = F_k^* \end{array}\right\} \begin{array}{c} \left\{\begin{array}{l} z_{k-1}+\frac{1}{2}h_{k-1} \\ z_1 \text{ if } k=1 \end{array}\right\} < z < \left\{\begin{array}{l} z_k+\frac{1}{2}h_k \\ z_{K+1} \text{ if } k=K+1 \end{array}\right\}; \quad k=1 \text{ to } K+1. \\ 0 \qquad \text{otherwise} \end{array}$$

$$\left.\begin{array}{l} V(z) = G_k \\ \\ V^*(z) = G_k^* \end{array}\right\} \begin{array}{c} z_k < z < z_k + h_k; \quad k = 1 \text{ to } K. \\ 0 \qquad \text{otherwise} \end{array} \tag{2.14}$$

The forms of these trial functions are illustrated in Figure 2.1.

Inserting these trial functions into variation equation 2.8 results in the equation

$$\begin{aligned} &\delta F_1^{*T}\left\{\int_0^{\frac{1}{2}} \mathbb{A}_1 F_1 h_1\, dx + G_1 - G_0\right\} \\ &+ \sum_{k=2}^{K} \delta F_k^{*T}\left\{\int_{\frac{1}{2}}^{1} \mathbb{A}_{k-1} F_k h_{k-1}\, dx + \int_0^{\frac{1}{2}} \mathbb{A}_k F_k h_k\, dx + G_k - G_{k-1}\right\} \\ &+ \delta F_{K+1}^{*T}\left\{\int_{\frac{1}{2}}^{1} \mathbb{A}_K F_{K+1} h_K\, dx + G_{K+1} - G_K\right\} \\ &+ \sum_{k=1}^{K} \delta G_k^{*T}\left\{\int_0^1 \mathbb{D}_k^{-1} G_k h_k\, dx + F_{k+1} - F_k\right\} = 0 \end{aligned} \tag{2.15}$$

---

†Shifting the domain of definition of the trial functions results in other approximation schemes with equivalently averaged nuclear constants.[34]

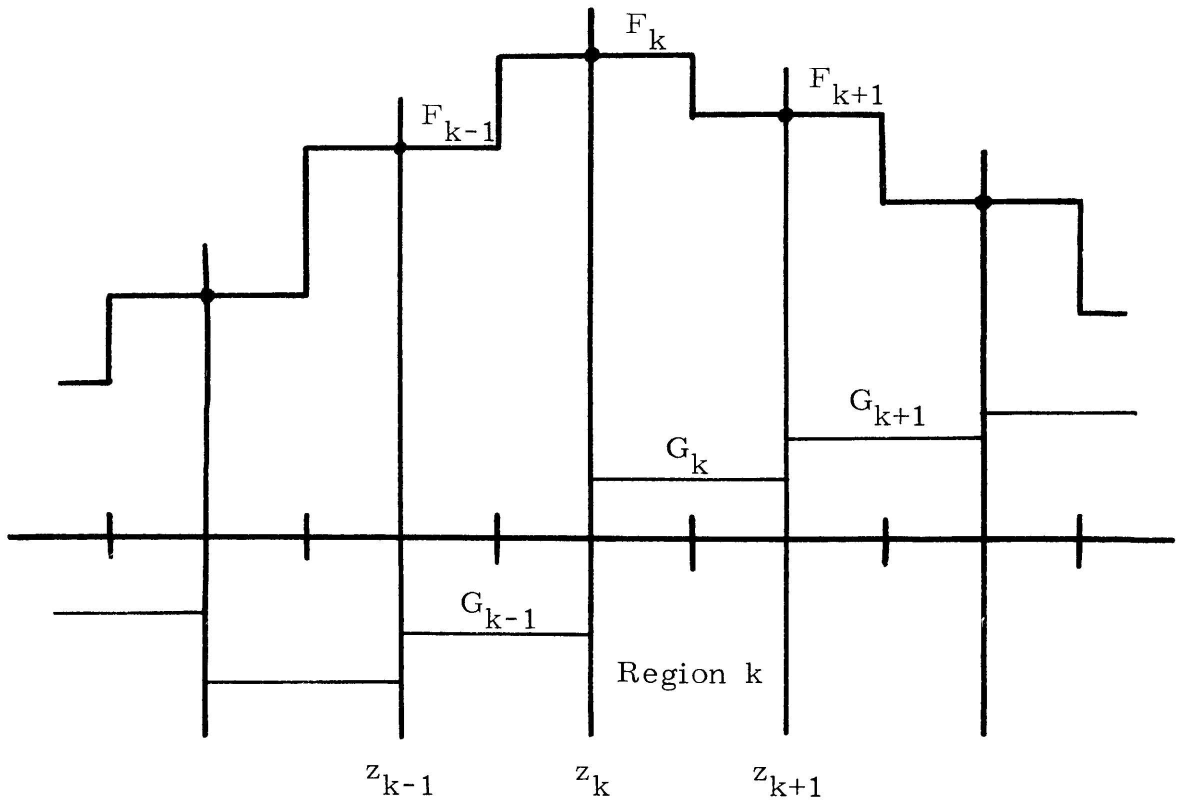


Figure 2.1. Conventional Nodal Finite Difference Approximation Trial Function Forms

Independent variation of all F_k^* and G_k^* then results in a system of 2K+1 equations and 2K+3 unknowns (including G_0 and G_{K+1}). The choice of boundary conditions supplies the missing equations. Zero flux boundary conditions can be imposed by setting $F_1 = F_{K+1} = 0$, which also requires $\delta F_1^* = \delta F_{K+1}^* = 0$ thereby eliminating G_0 and G_{K+1}, and results in a system of 2K-1 equations and 2K-1 unknowns. Symmetry boundary conditions can be imposed on the left by $G_0 = -G_1$ and on the right by $G_{K+1} = -G_K$, resulting in a system of 2K+1 equations and 2K+1 unknowns.

Elimination of all G_k, k = 1 to K, results in the standard three-point difference equations

$$b_1 F_1 + c_1 F_2 = 0 \qquad (2.16a)$$

$$a_k F_{k-1} + b_k F_k + c_k F_{k+1} = 0 \,; \quad k = 2 \text{ to } K \qquad (2.16b)$$

$$a_{K+1} F_K + b_{K+1} F_{K+1} = 0 \qquad (2.16c)$$

where Eqs. 2.16a and 2.16c are used for the cases of symmetry boundary conditions. The G × G matrix coefficients $\{a_k, b_k, c_k\}$ are of the form $A - \frac{1}{\lambda} B$ and are defined assuming homogeneous regional nuclear constants in section 1 of Appendix B. The matrix form of Eqs. 2.16 for the use of zero flux boundary conditions on the left and symmetry on the right is illustrated in Figure 2.2.

2.3.2 Multichannel Polynomial Synthesis

The one-dimensional neutron flux $\Phi_k(z)$ defined as nonzero only within each region k for each region (k=1 to K) can be approximated within each region as a polynomial of order N by the power series

$$\begin{bmatrix} \beta_2 & c_2 & & & \\ \ddots & \ddots & \ddots & & \\ & a_k & \beta_k & c_k & \\ & & \ddots & \ddots & \ddots \\ & & & a_{K+1} & \beta_{K+1}\end{bmatrix} \begin{bmatrix} F_2 \\ \vdots \\ F_k \\ \vdots \\ F_{K+1}\end{bmatrix} = \frac{1}{\lambda}\begin{bmatrix}\epsilon_2 & & & & \\ & \ddots & & & \\ & & \epsilon_k & & \\ & & & \ddots & \\ & & & & \epsilon_{K+1}\end{bmatrix}\begin{bmatrix}F_2\\ \vdots \\ F_k \\ \vdots \\ F_{K+1}\end{bmatrix}$$

Figure 2.2. Matrix Form of the Conventional Finite Difference Equations. Boundary conditions chosen are zero flux on the left and symmetry on the right.

$$a_k\,F_{k-1} + b_k\,F_k + c_k\,F_{k+1} = 0; \quad k = 2 \text{ to } K.$$

$$a_{K+1}\,F_K + b_{K+1}\,F_{K+1} = 0.$$

where: $\quad b_k = \beta_k - \frac{1}{\lambda}\,\epsilon_k; \quad k = 2 \text{ to } K+1.$

$$U_k^{(N)}(z) = \sum_{i=0}^{N} a_{k,i}\, x^i \tag{2.17}$$

where the distinction between z and x is understood since $0 \leqslant x \leqslant 1$ within each region k. Such approximations are not useful in diffusion theory because: (1) the resulting matrix equations relating the $a_{k,i}$'s contain full matrices similar to Hibert matrices which may be very difficult to solve; and (2) such matrices are almost always highly singular and may produce numerical instabilities in the solution method. These difficulties can be eliminated by employing polynomials in the trial functions in the following form:

$$U_k^{(N)}(z) \doteq \sum_{i=0}^{N} p_i^{(N)}(x) F_{k+\frac{i}{N}} \tag{2.18}$$

where the $p_i^{(N)}(x)$ are polynomials in x of degree N. This form is convenient because for a particular selection of the $p_i^{(N)}(x)$ the unknowns $F_{k+\frac{i}{N}}$ can be defined as the approximate flux solution evaluated at points $z_i + \frac{i}{N}$ within region k. For high order approximations, $i > 0$, the flux can be made continuous by imposing the following restrictions on $p_i^{(N)}(x)$:

$$p_i^{(N)}\left(\frac{\ell}{N}\right) = \begin{Bmatrix} 1 & \ell = i \\ 0 & \ell \neq i \end{Bmatrix} \text{ for } \ell = 0 \text{ to } N \tag{2.19}$$

The specific polynomial flux approximations of this form through degree N=3 are given below:

$$U_k^{(0)}(x) = F_k \tag{2.20a}$$

$$U_k^{(1)}(x) = (1-x)F_k + x\,F_{k+1} \tag{2.20b}$$

$$U_k^{(2)}(x) = (1-3x+2x^2)F_k + (4x-4x^2)F_{k+\frac{1}{2}} + (-x+2x^2)F_{k+1} \tag{2.20c}$$

$$U_k^{(3)}(x) = \left(1 - \frac{11}{2}x + 9x^2 - \frac{9}{2}x^3\right)F_k + \left(9x - \frac{45}{2}x^2 + \frac{27}{2}x^3\right)F_{k+\frac{1}{3}} + \left(-\frac{9}{2}x + 18x^2 - \frac{27}{2}x^3\right)F_{k+\frac{2}{3}} + \left(x - \frac{9}{2}x^2 + \frac{9}{2}x^3\right)F_{k+1} \tag{2.20d}$$

An immediate drawback of these approximations lies in the definitions of the corresponding current trial functions. Given a flux polynomial approximation of degree N, polynomial approximations for the current can be of order zero through N, and may even be of higher order than the flux approximation. Each set of chosen trial function pairs ultimately results in a characteristic complex band-structured matrix problem which may or may not have desirable numerical solution properties and is usually very difficult to solve.

Such problems can be eliminated by noting that the use of variational analysis attempts to force the current approximation to obey Fick's law. The obvious solution is direct use of Fick's law in the trial function forms

$$V_k(z) = -\mathbb{D}_k(z)\frac{d}{dz}U_k(z) \tag{2.21}$$

which results in simple band-structured matrix equations relating only flux unknowns. The use of current polynomial approximations of order N-1 as given in Eqs. 2.20 with flux approximations of order N, however, does not improve the situation.

The accuracy of these difference equations can be found first by eliminating all non-integer subscripted unknowns, then expanding the resulting three-point difference equations in a Taylor series about node k, and comparing results to the exact three-point difference solution known for the one-dimensional case.[7,35] By comparison of terms containing equal powers of h_k, it can be shown that the N=1 and N=2 polynomial approximations are accurate to order $\theta(h^2)$ while the N=3 approximation is accurate to order $\theta(h^3)$.

The approximation of a function by a polynomial of order N leads immediately to the concept of basis functions. The N+1 polynomial functions which multiply the N+1 unknowns in Eq. 2.18 form a basis for the approximation and can be called basis functions. The simplicity of basis functions becomes apparent in an error analysis of the approximation as follows. An approximate solution $U^{(N)}(z)$ of order N to the exact one-dimensional solution $\Phi(z)$ can be expressed as

$$U^{(N)}(z) = \sum_{k=1}^{K} \Phi(z_k)\,\Omega_k^{(N)}(z) \tag{2.22a}$$

where $\Omega_k^{(N)}(z)$ is a basis function of order N <u>centered</u> <u>about</u> node z_k. By Taylor series expansion about any node, it can be shown[36] that <u>if</u> $\Omega_k(z)$ satisfies:

$$\sum_{k=1}^{K} z_k^{\alpha}\,\Omega_k^{(N)}(z) = \left(\frac{z}{h_k}\right)^{\alpha} \quad \text{for } |\alpha| \leq N \tag{2.22b}$$

<u>then</u> $U^{(N)}(z)$ is an approximation to $\Phi(z)$ accurate to order $\theta(h_k^{N+1})$.

Basis functions found using Eqs. 2.22 are unique for each N and generally extend over surrounding regions. The forms of the basis functions for $N \leq 3$ are summarized below and illustrated in Figure 2.3. Since the following basis functions are symmetric, only the right half, $z \geq z_k$, is expressly given.

$$N = 0: \qquad \Omega_k^{(0)}(z) = \begin{cases} 1 & z_k \leq z < z_k + \frac{1}{2}h_k \\ 0 & \text{otherwise} \end{cases} \tag{2.23a}$$

$$N = 1: \qquad \Omega_k^{(1)}(z) = \begin{cases} (1-x) & z_k \leq z \leq z_{k+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.23b}$$

$$N = 2: \qquad \Omega_k^{(2)}(z) = \begin{cases} \frac{3}{4} - x^2 & z_k \leq z \leq z_k + \frac{1}{2}h_k \\ \frac{1}{2}\left(\frac{3}{2} - x\right)^2 & z_k + \frac{1}{2}h_k \leq z \leq z_{k+1} \\ \frac{1}{2}\left(\frac{1}{2} - x\right)^2 & z_{k+1} \leq z \leq z_{k+1} + \frac{1}{2}h_{k+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.23c}$$

$$N = 3: \qquad \Omega_k^{(3)}(z) = \begin{cases} \frac{1}{36}(30-54x^2+28x^3) & z_k \leq z \leq z_{k+1} \\ \frac{1}{36}(4-24x+30x^2-11x^3) & z_{k+1} \leq z \leq z_{k+2} \\ \frac{1}{36}(-1+3x-3x^2+x^3) & z_{k+2} \leq z \leq z_{k+3} \\ 0 & \text{otherwise} \end{cases} \tag{2.23d}$$

where $0 \leq x \leq 1$ within each region k in the above cases.

Use of these basis functions results in approximate solutions which are continuous for $N \geq 1$ and whose derivatives $dU^{(N)}(z)/dz$ through $d^{N-1}U^{(N)}(z)/dz^{N-1}$ are also continuous. In high order approximations in diffusion theory, it is advantageous to retain flux and current continuity and employ basis functions defined over two adjacent regions in order to produce three-point difference equations. This can be accomplished in the N=3 approximations with the cubic Hermite basis functions.

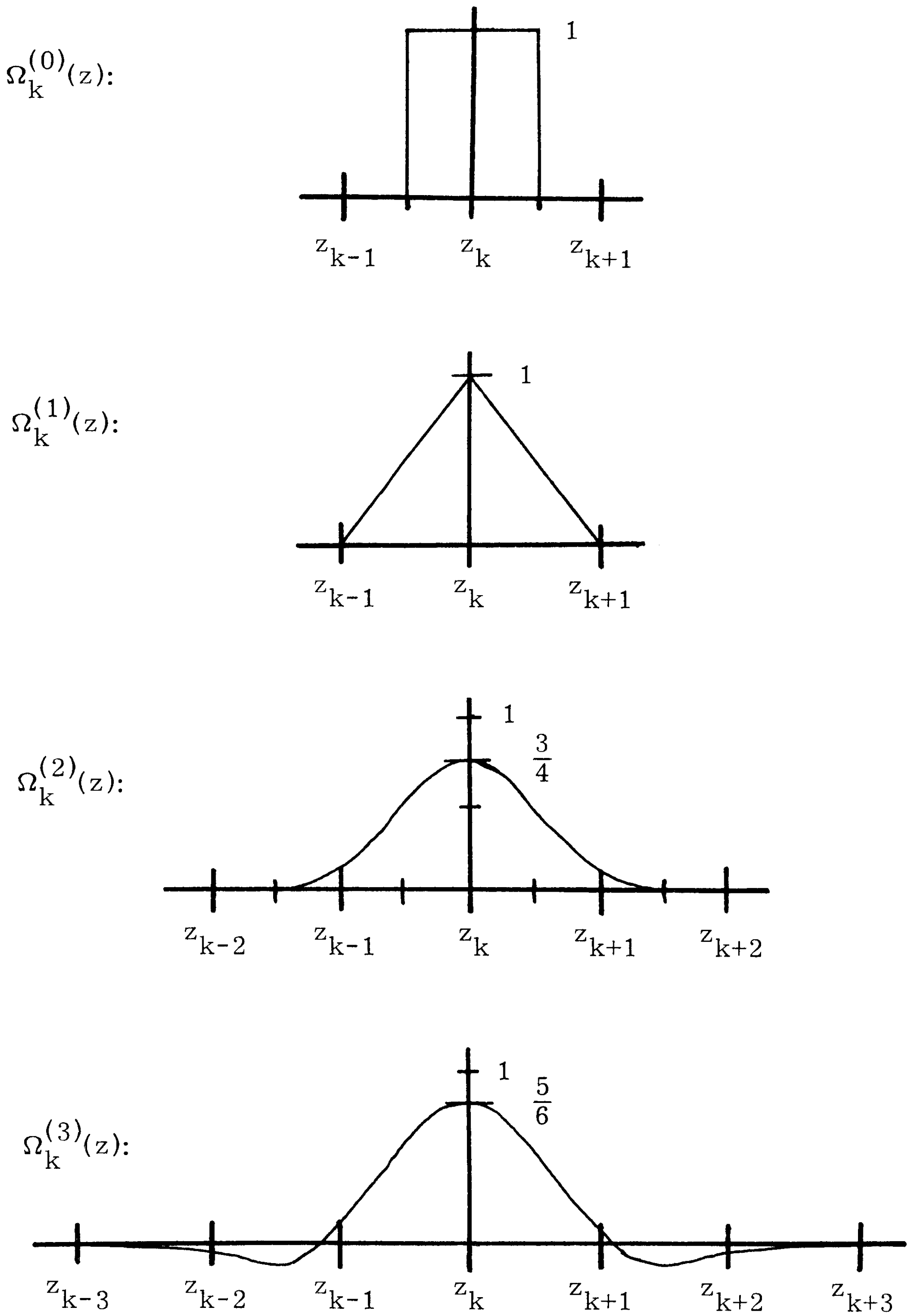


Figure 2.3. Basis Functions of Eqs. 2.23 for N = 0, 1, 2, and 3

The above cubic basis function $\Omega_k^{(3)}(z)$ can be constructed from a combination of either cubic B splines, $\Omega_k^B(z)$, or cubic Hermite polynomials, $\Omega_k^{H_1}(z)$ and $\Omega_k^{H_2}(z)$, as follows: [37]

$$\Omega_k^{(3)}(z) = -\frac{1}{6}\,\Omega_{k-1}^B(z) + \frac{4}{3}\,\Omega_k^B(z) - \frac{1}{6}\,\Omega_{k+1}^B(z) \qquad (2.24)$$

where:

$$\Omega_k^B(z) = \frac{2}{3}\,\Omega_k^{H_1}(z) + \frac{1}{6}\,\Omega_{k+1}^{H_1}(z) - \frac{1}{2}\,\Omega_{k+1}^{H_2}(z) \qquad (2.25)$$

The forms of these cubic B and Hermite polynomials are given below and illustrated in Figures 2.5 and 2.6. Again, only the right half of the functions are expressly given as Ω_k^B and $\Omega_k^{H_1}$ are symmetric, while $\Omega_k^{H_2}$ is antisymmetric.

$$\Omega_k^B(z) = \begin{cases} \frac{2}{3} - x^2 + \frac{1}{2}x^3 & z_k \leq z \leq z_{k+1} \\ \frac{1}{6}(1-x)^3 & z_{k+1} \leq z \leq z_{k+2} \\ 0 & \text{otherwise} \end{cases} \qquad (2.26)$$

$$\Omega_k^{H_1}(z) = \begin{cases} 1 - 3x^2 + 2x^3 & z_k \leq z \leq z_{k+1} \\ 0 & \text{otherwise} \end{cases} \qquad (2.27a)$$

$$\Omega_k^{H_2}(z) = \begin{cases} x - 2x^2 + x^3 & z_k \leq z \leq z_{k+1} \\ 0 & \text{otherwise} \end{cases} \qquad (2.27b)$$

where again $0 \leq x \leq 1$ in each region k.

The fact that the cubic Hermite polynomials form a basis for the cubic basis functions and extend over only two adjacent regions makes them very attractive for use in diffusion theory approximation methods.

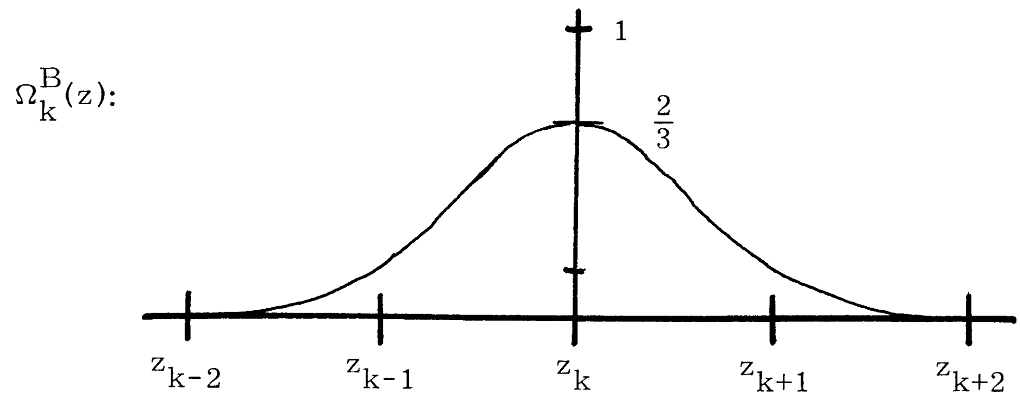


Figure 2.4. Cubic B Spline $\Omega_k^B(z)$ of Eq. 2.26

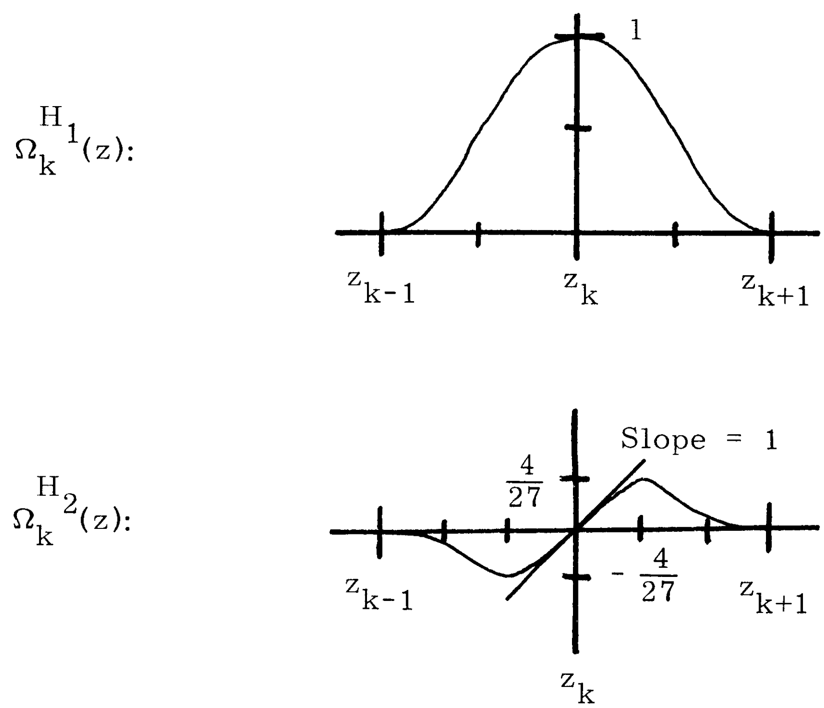


Figure 2.5. Cubic Hermite Basis Functions $\Omega_k^{H_1}(z)$ and $\Omega_k^{H_2}(z)$ of Eqs. 2.27

2.3.3 The Linear Basis Function Approximation

The group flux trial functions defined as nonzero within each region k can be expressed in modal-nodal form in terms of linear basis functions as

$$U_k(z) = (1-x)F_k + x\,F_{k+1} \qquad (2.28a)$$

$$; \quad k = 1 \text{ to } K.$$

$$U^*_k(z) = (1-x)F^*_k + x\,F^*_{k+1} \qquad (2.28b)$$

where F_k is the approximate group flux column vector at node z_k and $0 \leq x \leq 1$ with each region k. Although the flux trial functions are continuous, the current trial functions defined within each region by Eqs. 2.10 are not, and are given by

$$V_k(z) = \frac{1}{h_k}\,\mathbb{D}_k(x)\,[\,F_k - F_{k+1}\,] \qquad (2.28c)$$

$$; \quad k = 1 \text{ to } K.$$

$$V^*_k(z) = \frac{1}{h_k}\,\mathbb{D}_k(x)\,[\,F^*_{k+1} - F^*_k\,] \qquad (2.28d)$$

Insertion of these trial function forms into variation equation 2.12a results in the equation

$$\sum_{k=1}^{K} h_k \int_0^1 \Big\{ [(1-x)\delta F^*_k + x\delta F^*_{k+1}]^T\, \boldsymbol{\Lambda}_k(x)[(1-x)F_k + xF_{k+1}]$$

$$+ [\,\delta F^*_k - \delta F^*_{k+1}\,]^T \frac{1}{h_k^2}\,\mathbb{D}_k(x)[\,F_k - F_{k+1}]\Big\}\, dx = 0 \qquad (2.29)$$

Allowing arbitrary variations in all F^*_k results in a system of K+1 equations and K+1 unknowns which can be written as:

$$b_1F_1 + c_1F_2 = 0 \qquad (2.30a)$$

$$a_kF_{k-1} + b_kF_k + c_kF_{k+1} = 0 \quad ; \quad k = 2, K. \qquad (2.30b)$$

$$a_{K+1}F_K + b_{K+1}F_{K+1} = 0 \qquad (2.30c)$$

where the $G \times G$ matrix coefficients $\{a_k, b_k, c_k\}$ are of the form $A - \frac{1}{\lambda} B$ and are defined assuming homogeneous regional nuclear constants in section 2 of Appendix B. Zero flux boundary conditions can be imposed by use of only Eq. 2.30b with $F_1 = F_{K+1} = 0$, while symmetry boundary conditions require the use of the other equations as well. The matrix form of these equations for the boundary conditions of zero flux on the left and symmetry on the right is given in Figure 2.6.

2.3.4 The Cubic Hermite Basis Function Approximation[38,39]

The cubic Hermite polynomials can be incorporated into modal-nodal flux trial functions which allow continuous flux and continuous current by defining the flux trial functions within each region k as

$$U_k(z) = (1-3x^2+2x^3)F_k + (3x^2-2x^3)F_{k+1} + (-x+2x^2-x^3)\frac{\theta}{h_k}\mathbb{D}_k^{-1}(x)G_k + (x^2-x^3)\frac{\theta}{h_k}\mathbb{D}_k^{-1}(x)G_{k+1} \qquad (2.31a)$$

$$U_k^*(z) = (1-3x^2+2x^3)F_k^* + (3x^2-2x^3)F_{k+1}^* + (-x+2x^2-x^3)\frac{\theta}{h_k}\mathbb{D}_k^{-1}(x)G_k^* + (x^2-x^3)\frac{\theta}{h_k}\mathbb{D}_k^{-1}(x)G_{k+1}^* \qquad (2.31b)$$

where $k = 1$ to K.

F_k is again the approximate group flux solution vector at node z_k, and G_k is proportional to the approximate group current solution vector at node z_k. Application of Fick's law defines the current trial functions for each k as

$$
\begin{bmatrix}
\beta_2 & \gamma_2 & & & & \\
\ddots & \ddots & \ddots & & & \\
 & \alpha_k & \beta_k & \gamma_k & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & & \alpha_{K+1} & \beta_{K+1}
\end{bmatrix}
\begin{bmatrix}
F_2 \\ | \\ F_k \\ | \\ F_{K+1}
\end{bmatrix}
= \frac{1}{\lambda}
\begin{bmatrix}
\epsilon_2 & \zeta_2 & & & & \\
\ddots & \ddots & \ddots & & & \\
 & \delta_k & \epsilon_k & \zeta_k & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & & \delta_{K+1} & \epsilon_{K+1}
\end{bmatrix}
\begin{bmatrix}
F_2 \\ | \\ F_k \\ | \\ F_{K+1}
\end{bmatrix}
$$

Figure 2.6. Matrix Form of the Linear Finite Element Method Approximation. Eqs. 2.35 for the case of zero flux on the left and symmetry boundary conditions on the right.

where:

$$
\left.
\begin{aligned}
a_k &= \alpha_k - \frac{1}{\lambda}\,\delta_k \\
b_k &= \beta_k - \frac{1}{\lambda}\,\epsilon_k \\
c_k &= \gamma_k - \frac{1}{\lambda}\,\zeta_k
\end{aligned}
\right\}
\quad k = 2 \text{ to } K+1.
$$

$$V_k(z) = \frac{1}{h_k} \mathbb{D}_k(x)(6x-6x^2)[F_{k+1}-F_k]$$
$$+ (1-4x+3x^2)\theta G_k + (-2x+3x^2)\theta G_{k+1} \qquad (2.31c)$$

$$V_k^*(z) = \frac{1}{h_k} \mathbb{D}_k(x)(6x-6x^2)[F_k^*-F_{k+1}^*]$$
$$+ (-1+4x-3x^2)\theta G_k^* + (2x-3x^2)\theta G_{k+1}^* \qquad (2.31d)$$

Continuity of flux and current are automatically guaranteed since

$$\begin{aligned} U_k(0) &= U_{k-1}(h_{k-1}) = F_k \\ U_k^*(0) &= U_{k-1}^*(h_{k-1}) = F_k^* \\ V_k(0) &= V_{k-1}(h_{k-1}) = \theta G_k \\ V_k^*(0) &= V_{k-1}^*(h_{k-1}) = -\theta G_k^{*\dagger} \end{aligned} \qquad (2.32)$$

The normalization constant θ is introduced in order to produce stiffness matrices having small condition numbers and can be chosen such that $\frac{\theta}{D_k(0)} \simeq 1$.

Insertion of these trial function forms into variation equation 2.12a results in a lengthy equation which can be written as follows:

---

†Such a choice of $-G_k^*$ allows the matrix of coefficients to be positive definite. Cf., Chapter 4.

$$
\begin{aligned}
&\delta F_1^{*T}\{b1_1 F_1 + b2_1 G_1 + c1_1 F_2 + c2_1 G_2\} \\
&\delta G_1^{*T}\{b3_1 F_1 + b4_1 G_1 + c3_1 F_2 + c4_1 G_2\} \\
+ \sum_{k=2}^{K} &\delta F_k^{*T}\{a1_k F_{k-1} + a2_k G_{k-1} + b1_k F_k + b2_k G_k + c1_k F_{k+1} + c2_k G_{k+1}\} \\
+ \sum_{k=2}^{K} &\delta G_k^{*T}\{a3_k F_{k-1} + a4_k G_{k-1} + b3_k F_k + b4_k G_k + c3_k F_{k+1} + c4_k G_{k+1}\} \\
&\delta F_{K+1}^{*T}\{a1_{K+1} F_K + a2_{K+1} G_K + b1_{K+1} F_{K+1} + b2_{K+1} G_{K+1}\} \\
&\delta G_{K+1}^{*T}\{a3_{K+1} F_K + a4_{K+1} G_K + b3_{K+1} F_{K+1} + b4_{K+1} G_{K+1}\} = 0.
\end{aligned}
\tag{2.33}
$$

where the $G \times G$ matrix coefficients $\{a1, \ldots, c4\}$ are of the form $A - \frac{1}{\lambda} B$ and are defined assuming homogeneous regional nuclear constants in section 3 of Appendix B.

The choice of either zero flux, $F_k = 0$ as well as $\delta F_k^* = 0$, or zero current, $G_k = 0$ as well as $\delta G_k^* = 0$, boundary conditions for $k = 1$ or $K+1$ along with arbitrary variations of the remaining F_k^* and G_k^* results in a system of 2K equations and 2K unknowns. Figure 2.7 illustrates the matrix form of such a system for the case of zero flux on the left and zero current on the right boundary conditions.

The basis functions and approximation techniques presented in this section are applied to the proposed approximation methods in the next chapter. Also, various techniques for treating zero flux and symmetry boundary conditions are discussed. The matrix properties of the equations resulting from the above finite element approximations and their solution methods are discussed in Chapter 4.

$$
\begin{bmatrix}
b4_1 & c3_1 & c4_1 & & & & & & & & & \\
a2_2 & b1_2 & b2_2 & c1_2 & c2_2 & & & & & & & \\
a4_2 & b3_2 & b4_2 & c3_2 & c4_2 & & & & & & & \\
 & & \ddots & & \ddots & & \ddots & & & & & \\
 & & & a1_k & a2_k & b1_k & b2_k & c1_k & c2_k & & & \\
 & & & a3_k & a4_k & b3_k & b4_k & c3_k & c4_k & & & \\
 & & & & & & \ddots & & \ddots & & \ddots & \\
 & & & & & & & a1_K & a2_K & b1_K & b2_K & c1_K \\
 & & & & & & & a3_K & a4_K & b3_K & b4_K & c3_K \\
 & & & & & & & & & a1_{K+1} & a2_{K+1} & b1_{K+1}
\end{bmatrix}
\begin{bmatrix}
G_1 \\ F_2 \\ G_2 \\ \vdots \\ F_k \\ G_k \\ \vdots \\ F_K \\ G_K \\ F_{K+1}
\end{bmatrix}
= 0
$$

Figure 2.7. Matrix Form of the Cubic Hermite Finite Element Method Approximation. Eqs. 2.39 for the case of zero flux on the left and symmetry boundary conditions on the right.

where:
$$
\left.
\begin{aligned}
an_k &= \alpha n_k - \frac{1}{\lambda}\,\delta n_k \\
bn_k &= \beta n_k - \frac{1}{\lambda}\,\epsilon n_k \\
cn_k &= \gamma n_k - \frac{1}{\lambda}\,\zeta n_k
\end{aligned}
\right\}
\quad
\begin{aligned}
&n = 1 \text{ to } 4; \\
&\text{and} \\
&k = 1 \text{ to } K+1.
\end{aligned}
$$

Chapter 3
DEVELOPMENT OF A CONSISTENT COARSE MESH APPROXIMATION METHOD

3.1 Formulation

The finite element methods have been shown[32,33] to approximate accurately flux solutions and criticality measurements of multigroup diffusion theory when applied to problems allowing homogeneous nuclear material within the mesh regions. Use of such homogeneous material, while simplifying the calculation of the matrix elements (since numerical integrations are not required), may result in limiting the region mesh sizes allowed unless some type of homogenization procedure is used. If the mesh spacing is chosen such that some or all mesh regions are heterogeneous, then direct application of the variational techniques given in Chapter 2 results in weight averaging the nuclear constants with products of the basis functions and their derivatives, as given by the approximation. Although such a procedure is a direct application of the finite element technique, the accuracy of such methods depends upon the placement of the mesh regions and may vary significantly as their placement is altered.

A more useful homogenization procedure which is commonly used in reactor diffusion theory analysis allows the nuclear material within each mesh region to be homogenized by flux weighting with an assumed flux shape determined *a priori* within that region in order (hopefully) to preserve reaction rates.

In large reactors the core can be thought of as composed of a lattice of heterogeneous fuel subassemblies containing fuel, clad, coolant channels, and/or absorption control rods. Each subassembly can be divided into several distinct homogeneous regions whose few-group microcell macroscopic nuclear constants are found by multigroup energy-dependent calculations.[40] Detailed subassembly solutions, $\psi_k(r)$, are then found for each subassembly k by assuming that the current on the boundary of the subassemblies is zero. Flux weighting the nuclear material in each subassembly with the corresponding detailed subassembly solution for each subassembly region then results in regional homogeneous nuclear constants $\langle \Sigma_k \rangle$ which may better approximate the physics of the region.

$$\langle \Sigma_k \rangle = \frac{\int_k \psi_k(r)\, \Sigma_k(r)\, dr}{\int_k \psi_k(r)\, dr} \tag{3.1}$$

Proper use of detailed flux weighted constants can lead to accurate criticality measurements, but the detailed *a priori* fine flux structure within each region is lost since it appears only in cross-section homogenization and not in the approximation. Attempts to retain the fine flux structure have only recently been proposed in several multichannel synthesis approximations.[27,41,42,43] Unfortunately, each of these approximations are approximations in themselves and do not reduce to desirable approximations if the detailed flux solutions are themselves constant, as would be the case in large homogeneous regions.

Just as the discontinuous multichannel synthesis approximation method can be shown to reduce to low order difference equations (of the type which could result using the finite element method with constant or flat basis functions) when constant trial functions are used, approximation methods are presented below which retain the given detailed flux structure and also reduce to the higher order finite element approximations. The use of linear or cubic Hermite basis functions in the approximation provides flux continuity and results in better approximation accuracy.

The approximations are presented and discussed for the case of one-dimensional, multigroup diffusion theory. Extension to higher dimensions remains a problem that will require some further study. The approximations which are the linear basis functions are considered in the next section, while the approximations using the cubic Hermite basis functions are considered in section 3.3.

3.2 The Proposed Linear Basis Function Approximations

The proposed approximation method utilizing linear basis functions and defined as nonzero within each mesh region k, $k = 1$ to K, is given by the following modal-nodal trial function forms:

$$U_k(z) = \psi_k(x)\left[\psi_k^{-1}(0)(1-x)F_k + \psi_k^{-1}(1)x\, F_{k+1}\right] \tag{3.2a}$$

$$U_k^*(z) = \psi_k^*(x)\left[\psi_k^{*-1}(0)(1-x)F_k^* + \psi_k^{*-1}(1)\, xF_{k+1}^*\right] \tag{3.2b}$$

$$V_k(z) = \eta_k(x)\left[\psi_k^{-1}(0)(1-x)F_k + \psi_k^{-1}(1)\, xF_{k+1}\right]$$

$$+ \frac{1}{h_k}\mathbb{D}_k(x)\,\psi_k(x)\left[\psi_k^{-1}(0)F_k - \psi_k^{-1}(1)\,F_{k+1}\right] \tag{3.2c}$$

$$V_k^*(z) = \eta_k^*(x)\,[\,\psi_k^{*^{-1}}(0)(1-x)F_k^* + \psi_k^{*^{-1}}(1)\,x\,F_{k+1}^*\,]$$
$$+ \frac{1}{h_k}\,\mathbb{D}_k(x)\psi_k^*(x)\,[\,\psi_k^{*^{-1}}(1)\,F_{k+1}^* - \psi_k^{*^{-1}}(0)\,F_k^*\,] \qquad (3.2d)$$

where:

$$x = (z-z_k)/h_k \qquad (3.2e)$$

and $0 \leqslant x \leqslant 1$, as $z_k \leqslant z \leqslant z_{k+1}$, for each region k = 1 to K.

F_k is the unknown approximate group flux column vector at node z_k, and ψ_k, ψ_k^*, η_k, and η_k^* are G × G diagonal matrices composed of the detailed group flux $\psi_{g,k}(z)$ and group current $\eta_{g,k}(z)$ solutions, and their adjoints, defined as nonzero only within region k. Because of the variable transformation between z and x, $\psi_k(0)$ represents $\psi_k(z_k)$, and $\psi_k(1)$ represents $\psi_k(z_{k+1})$; neither of which, for the moment, is allowed to be zero for any region. The detailed current solutions are given from the detailed flux solutions by Fick's law as

$$\eta_k(z) = -\mathbb{D}_k(z)\,\frac{d\psi_k(z)}{dz} \qquad (3.3a)$$

$$\eta_k^*(z) = +\mathbb{D}_k(z)\,\frac{d\psi_k^*(z)}{dz} \qquad (3.3b)$$

As a result, the current trial functions are related to the flux trial functions by analogous expressions.

Continuity of the flux is imposed by the form of the trial functions since

$$U_k(0) = U_{k-1}(h_{k-1}) = F_k \qquad (3.4a)$$

$$U_k^*(0) = U_{k-1}^*(h_{k-1}) = F_k^* \qquad (3.4b)$$

The current trial functions, however, are discontinuous. It is evident by comparison to Eqs. 2.28, that this approximation reduces to the linear basis function finite element method if the detailed flux solutions for each group are taken to be constant.

Insertion of these trial function forms in Eq. 2.12a results in the following variation equation:

$$\sum_{k=1}^{K} h_k \int_0^1 \Bigg\{ \psi_k^{*T}(x)\psi_k^{*T^{-1}}(0)(1-x)\delta F_k^{*T} \boldsymbol{\Lambda}_k(x)U_k(x)$$

$$+ \psi_k^{*T}(x)\psi_k^{*T^{-1}}(1)\, x\, \delta F_{k+1}^{*T} \boldsymbol{\Lambda}_k(x)\, U_k(x)$$

$$+ \left[\eta_k^{*T}(x)\,\psi_k^{*T^{-1}}(0)\,(1-x) - \frac{1}{h_k}\, \mathbb{D}_k(x)\psi_k^{*T}(x)\psi_k^{*T^{-1}}(0) \right] \delta F_k^{*T}\, \mathbb{D}_k^{-1}(x)V_k(x)$$

$$+ \left[\eta_k^{*T}(x)\psi_k^{*T^{-1}}(1)x + \frac{1}{h_k}\, \mathbb{D}_k(x)\psi_k^{*T}(x)\psi_k^{*T^{-1}}(1) \right] \delta F_{k+1}^{*T}\, \mathbb{D}_k^{-1}(x)V_k(x) \Bigg\} dx = 0 \tag{3.5}$$

This equation can be written in the form

$$\delta F_1^{*T}[b_1F_1 + c_1F_2]$$

$$+ \sum_{k=2}^{K} \delta F_k^{*T}[a_kF_{k-1} + b_kF_k + c_kF_{k+1}]$$

$$+ \delta F_{K+1}^{*T}[a_{K+1}F_K + b_{K+1}F_{K+1}] = 0 \tag{3.6}$$

where the $G \times G$ matrix coefficients $\{a_k, b_k, c_k\}$ are integral quantities of the form $A - \frac{1}{\lambda}B$ and are defined in detail in section 1 of Appendix C.

External zero flux boundary conditions are easily imposed by setting $F_1 = F_{K+1} = 0$. This requires that F_1^* and F_{K+1}^* must then also be zero, which in turn requires the δF_1^* and δF_{K+1}^* coefficients in

Eqs. 3.5 and 3.6 to vanish. Allowing independent variations in the remaining F_k^*, k = 2 to K, results in a matrix problem of the form illustrated in Figure 2.6 which would contain K-1 equations and K-1 unknowns.

Zero current boundary equations are found using symmetry considerations. If, for example, a zero current or symmetry boundary condition is imposed on the right at z_{K+1}, then a "boundary condition equation" can be derived by assuming a pseudo-region $k = K+1$ of width h_K having mirror image properties of region K about z_{K+1} with corresponding symmetric flux and antisymmetric current properties of the detailed flux and current solutions. These properties in pseudo-region K+1 can be related to properties of region K as a function of x in each region as

$$\mathbb{D}_{K+1}(x) = \mathbb{D}_K(1-x) \tag{3.7a}$$

$$\Lambda_{K+1}(x) = \Lambda_K(1-x) \tag{3.7b}$$

and

$$U_{K+1}(x) = U_K(1-x) \tag{3.8a}$$

$$U^*_{K+1}(x) = U^*_K(1-x) \tag{3.8b}$$

$$V_{K+1}(x) = -V_K(1-x) \tag{3.8c}$$

$$V^*_{K+1}(x) = -V^*_K(1-x) \tag{3.8d}$$

The addition of pseudo-region K+1 to the summation in Eq. 3.5 results in the calculation of coefficients a_{K+1} and b_{K+1} in Eq. 3.6. Detailed definitions of the $G \times G$ zero current coefficient matrices b_1, c_1, a_{K+1}, and b_{K+1}, all of which vanish for the case of zero flux boundary conditions, are also given in Appendix C.1. If symmetry is imposed on both sides of the problem, independent variations in F_k^* for k = 1 to K + 1 result in a matrix problem of K+1 equations and K+1 unknowns of similar form as illustrated in Figure 2.6.

Other boundary conditions may be imposed on the approximation, including albedo and reflector boundary conditions, which specify the flux to current ratio at the boundaries. Such conditions will always lead to a variation equation of the form of Eq. 3.6, where in general the matrix coefficients a_2, b_2, b_K, and c_K as well as the boundary coefficients b_1, c_1, a_{K+1}, and b_{K+1} will have modified definitions.

A serious drawback of the approximation given by Eqs. 3.2 is that it does not allow the use of detailed flux solutions containing explicit zero flux boundary conditions. For this reason the exact solution, $\psi_k(z) = \Phi_k(z)$ for all k, is excluded from the class of admissible trial function forms. However, such detailed solutions can be allowed by modifying the trial function forms in the boundary regions. If a detailed solution $\psi_1(z)$ is given in the first region with the zero flux condition $\psi_1(z_1) = 0$, for example, the trial functions of Eqs. 3.2 could be modified for region k=1 as

$$U_1(z) = \psi_1(x)\psi_1^{-1}(1)F_2 \tag{3.9a}$$

$$U_1^*(z) = \psi_1^*(x)\psi_1^{*-1}(1)F_2 \tag{3.9b}$$

$$V_1(z) = \eta_1(x)\psi_1^{-1}(1)F_2 \tag{3.9c}$$

$$V_1^*(z) = \eta_1^*(x)\psi_1^{*-1}(1)F_2^* \tag{3.9d}$$

In this way, the imposed zero flux boundary condition is explicitly given by $\psi_1(z)$ rather than in the form of the trial function. Similar trial functions can be given for an explicit zero flux boundary condition in the last region, k=K.

The use of these special trial functions in the boundary regions alters the definitions of the matrix coefficients b_2 and b_K as given in Eq. 3.6. Detailed definitions of these coefficients when these special trial functions are used are also included in Appendix C.

Regardless of the types of boundary conditions imposed, Eq. 3.6 results in an N $\times$ N matrix problem of the form

$$\mathbb{A}\underline{F} = \frac{1}{\lambda}\mathbb{B}\underline{F} \tag{3.10}$$

where $\mathbb{A}$ and $\mathbb{B}$ are independent of λ. The order N of the matrix equations is dependent upon the chosen boundary conditions, and is given for various choices in Table 3.1.

Table 3.1. Matrix Order N of the Proposed Linear Basis Function Approximations as a Function of the Imposed Boundary Conditions.

1 – Explicit or Implicit Zero Flux

2 – Symmetry

Boundary Condition Type		Matrix Order
on Left	on Right	N
1	1	$G \times (K-1)$
1	2	$G \times K$
2	1	$G \times K$
2	2	$G \times (K+1)$

3.3 The Proposed Cubic Hermite Basis Function Approximation

The proposed modal-nodal approximation method utilizing the cubic Hermite polynomials

$$\begin{aligned} p_1(x) &= 1 - 3x^2 + 2x^3 \\ p_2(x) &= 3x^2 - 2x^3 \\ p_3(x) &= -x + 2x^2 - x^3 \\ p_4(x) &= x^2 - x^3 \end{aligned} \tag{3.11}$$

and their negative derivatives

$$\begin{aligned} q_1(x) &= 6x - 6x^2 \\ q_2(x) &= -6x + 6x^2 \\ q_3(x) &= 1 - 4x + 3x^2 \\ q_4(x) &= -2x + 3x^2 \end{aligned} \tag{3.12}$$

and defined as nonzero only within each mesh region k, is given by the below regional trial function forms. F_k and G_k are again the unknown group column approximate flux and current solutions at z_k respectively, and the remaining symbols have been previously defined. As in the cubic Hermite finite element method described in section 2.3.4 of Chapter 2, θ is an optional normalization parameter.

$$\begin{aligned} U_k(z) = \psi_k(x)[\,&\psi_k^{-1}(0)p_1(x)F_k + \psi_k^{-1}(1)p_2(x)F_{k+1} \\ &+ h_k\theta \mathbb{D}_k^{-1}(0)\psi_k^{-1}(0)p_3(x)G_k + h_k\theta \mathbb{D}_k^{-1}(1)\psi_k^{-1}(1)p_4(x)G_{k+1}\,] \end{aligned} \tag{3.13a}$$

$$U_k^*(z) = \psi_k^*(x)\,[\,\psi_k^{*-1}(0)\,p_1(x)\,F_k^* + \psi_k^{*-1}(1)\,p_2(x)\,F_{k+1}^*$$

$$+ h_k \theta \mathbb{D}_k^{-1}(0) \psi_k^{*-1}(0)\,p_3(x) G_k^* + h_k \theta \mathbb{D}_k^{-1}(1) \psi_k^{*-1}(1)\,p_4(x) G_{k+1}^*\,] \qquad (3.13b)$$

$$V_k(z) = \eta_k(x)\,[\,\psi_k^{-1}(0)\,p_1(x)\,F_k + \psi_k^{-1}(1)\,p_2(x)\,F_{k+1}$$

$$+ h_k \theta \mathbb{D}_k^{-1}(0) \psi_k^{-1}(0)\,p_3(x) G_k + h_k \theta \mathbb{D}_k^{-1}(1)\,\psi_k^{-1}(1)\,p_4(x)\,G_{k+1}\,]$$

$$+ \mathbb{D}_k(x) \psi_k(x)\,[\,\frac{1}{h_k}\,\psi_k^{-1}(0)\,q_1(x) F_k + \frac{1}{h_k}\,\psi_k^{-1}(1)\,q_2(x) F_{k+1}$$

$$+ \theta \mathbb{D}_k^{-1}(0) \psi_k^{-1}(0)\,q_3(x) G_k + \theta \mathbb{D}_k^{-1}(1)\,\psi_k^{-1}(1)\,q_4(x) G_{k+1}\,] \qquad (3.13c)$$

$$V_k^*(z) = \eta_k^*(x)\,[\,\psi_k^{*-1}(0)\,p_1(x) F_k^* + \psi_k^{*-1}(1)\,p_2(x) F_{k+1}^*$$

$$+ h_k \theta \mathbb{D}_k^{-1}(0)\,\psi_k^{*-1}(0)\,p_3(x) G_k^* + h_k \theta \mathbb{D}_k^{-1}(1)\,\psi_k^{*-1}(1)\,p_4(x) G_{k+1}^*\,]$$

$$- \mathbb{D}_k(x) \psi_k^*(x)\,[\,\frac{1}{h_k}\,\psi_k^{*-1}(0)\,q_1(x) F_k^* + \frac{1}{h_k}\,\psi_k^{*-1}(1)\,q_2(x) F_{k+1}^*$$

$$+ \theta \mathbb{D}_k^{-1}(0) \psi_k^{*-1}(0)\,q_3(x) G_k^* + \theta \mathbb{D}_k^{-1}(1)\,\psi_k^{*-1}(1)\,q_4(x) G_{k+1}^*\,] \qquad (3.13d)$$

Again, for the moment, $\psi_k(0)$ and $\psi_k(1)$ and their adjoints are not allowed to be zero in any region.

The forms of these trial functions impose both flux and current continuity since:

$$U_k(z_k) = U_{k-1}(z_{k-1}+h_{k-1}) = F_k \qquad (3.14a)$$

$$U_k^*(z_k) = U_{k-1}^*(z_{k-1}+h_{k-1}) = F_k^* \qquad (3.14b)$$

$$V_k(z_k) = V_{k-1}(z_{k-1}+h_{k-1}) = G_k \qquad (3.14c)$$

$$V_k^*(z_k) = V_{k-1}^*(z_{k-1}+h_{k-1}) = -G_k^* \qquad (3.14d)$$

where it is assumed that at region mesh points the detailed current solutions are zero:

$$\eta_k(z_k) = \eta_k(z_k+h_k) = 0 \tag{3.15a}$$

$$\overset{*}{\eta}_k(z_k) = \overset{*}{\eta}_k(z_k+h_k) = 0 \tag{3.15b}$$

for all regions $k = 1$ to K.

Also, by comparison to Eqs. 2.31, it is evident that this approximation reduces to the cubic Hermite finite element method when the ψ_k's are constant, and the η_k's are correspondingly zero.

The insertion of the above trial function forms into variation equation 2.12a results in a lengthy equation which can be simplified to the following form:

$$\begin{aligned}
&\delta \overset{*}{F}{}_1^T \left[b1_1 F_1 + b2_1 G_1 + c1_1 F_2 + c2_1 G_2 \right] \\
&+ \delta \overset{*}{G}{}_1^T \left[b3_1 F_1 + b4_1 G_1 + c3_1 F_2 + c4_1 G_2 \right] \\
+ \sum_{k=2}^{K} &\left\{ \delta \overset{*}{F}{}_k^T \left[a1_k F_{k-1} + a2_k G_{k-1} + b1_k F_k + b2_k G_k + c1_k F_{k+1} + c2_k G_{k+1} \right] \right. \\
&\left. + \delta \overset{*}{G}{}_k^T \left[a3_k F_{k-1} + a4_k G_{k-1} + b3_k F_k + b4_k G_k + c3_k F_{k+1} + c4_k G_{k+1} \right] \right\} \\
&+ \delta \overset{*}{F}{}_{K+1}^T \left[a1_{K+1} F_K + a2_{K+1} G_K + b1_{K+1} F_{K+1} + b2_{K+1} G_{K+1} \right] \\
&+ \delta \overset{*}{G}{}_{K+1}^T \left[a3_{K+1} F_K + a4_{K+1} G_K + b3_{K+1} F_{K+1} + b4_{K+1} G_{K+1} \right] = 0
\end{aligned} \tag{3.16}$$

where the detailed definitions of the twelve integral $G \times G$ matrix coefficients $\{a1_k, \ldots, c4_k\}$ of the form $A - \frac{1}{\lambda} B$ for all k are given in section 2 of Appendix C.

Boundary conditions for either zero flux or symmetry are easily imposed by setting either F_k or G_k, respectively, to zero with $k = 1$ for the conditions on the left at z_1 or $k = K+1$ for conditions on the right at z_{K+1}. The corresponding variations for $k = 1$ and $k = K+1$ then vanish. Allowing arbitrary variations in the remaining F_k^* and G_k^* in Eq. 3.16 results in a system of 2K G $\times$ G matrix equations relating 2K G column vector unknowns, as illustrated in Figure 2.7.

Explicit zero flux boundary conditions imposed by $\psi_1(z_1) = 0$ or $\psi_K(z_{K+1}) = 0$ can be incorporated into the approximation by modifying the trial function definitions in the boundary regions. The modified flux trial functions in the first region, for example, are

$$U_1(x) = \psi_1(x)[\psi_1^{-1}(1)F_2 + h_1\theta\mathbb{D}_1^{-1}(0)\psi_1^{-1}(0)p_3(x)G_1 + h_1\theta\mathbb{D}_1^{-1}(1)\psi_1^{-1}(1)p_4(x)G_2]$$

$$U_1^*(x) = \psi_1^*(x)[\psi_1^{*-1}(1)F_2^* + h_1\theta\mathbb{D}_1^{-1}(0)\psi_1^{*-1}(0)p_3(x)G_1^* + h_1\theta\mathbb{D}_1^{-1}(1)\psi_1^{*-1}(1)p_4(x)G_2^*] \tag{3.17}$$

where the current trial functions are again given by Fick's laws. Use of modified trial function forms of this type in the boundary regions results in 2K equations with different definitions of $c3_1$, $a2_2$, $b1_2$, $b2_2$, and $b3_2$ as well as $b1_K$, $b2_K$, $c2_K$, and $a3_{K+1}$, which are also included in Appendix C.

Other boundary condition restrictions may be imposed on this approximation, but the matrix form of the resulting difference equations will remain unchanged. Only the coefficients defined for $k = 1, 2, K$, and $K+1$ will in general be altered.

The matrix equations resulting from this approximation can always be written as

$$\mathbb{A}\,\underline{F} = \frac{1}{\lambda}\,\mathbb{B}\,\underline{F} \tag{3.18}$$

where $\mathbb{A}$ and $\mathbb{B}$ are $(G \times 2K)$ by $(G \times 2K)$ matrices, independent of λ, and $\underline{F}$ is the $G \times K$ column vector of unknowns containing both F_k and G_k column vectors for each k. The matrix properties and solution methods of the matrix equations derived in these proposed approximation methods are discussed in the next chapter.

Chapter 4
NUMERICAL SOLUTION TECHNIQUES

The matrix properties of the difference equations resulting from both the proposed approximations and the finite element methods in one dimension, as well as the solution schemes used to solve these equations, are summarized in the following section. Various calculational and programming techniques used in conjunction with these approximation methods and their solution schemes are presented and discussed in section 4.2.

4.1 Solution Methods and Matrix Properties

The matrix equations which result from the approximations given in this report are of the form

$$\mathbb{A}\underline{F} = \frac{1}{\lambda}\mathbb{B}\,\underline{F} \tag{4.1}$$

and are solved using the fission source power iteration method without fission source renormalization.[7,44] The method of solution is illustrated schematically in Figure 4.1. Other definitions of the iteration eigenvalue $\lambda^{(i)}$ can be found elsewhere.[45]

Figure 4.1 illustrates that an outer iteration solution scheme[46] is used, and that the geometry and nuclear properties of the reactor are not altered. Since the fission source is not normalized by the iteration eigenvalue during the iterations, $\lambda^{(i)}$ converges to the effective multiplication factor, k_{eff}, of the problem. Had fission source renormalization been included, by $\underline{S}^{(i)} = \mathbb{B}\underline{F}^{(i)}/\lambda^{(i-1)}$ for example, then $\lambda^{(i)}$

$i = 0$

Guess $\underline{F}^{(1)} > 0$

$i = i + 1$ <u>Outer Iteration</u>

$\underline{S}^{(i)} = \mathbb{B}\, \underline{F}^{(i)}$

$\underline{\tilde{F}}^{(i+1)} = \mathbb{A}^{-1}\, \underline{S}^{(i)}$ <u>Matrix Inversion</u>

$$\lambda^{(i)} = \frac{(1, \mathbb{B}\, \underline{\tilde{F}}^{(i+1)})}{(1, \mathbb{B}\, \underline{F}^{(i)})}$$

$$\underline{e}^{(i)} = \frac{1}{\lambda}\, \underline{\tilde{F}}^{(i+1)} - \underline{F}^{(i)}$$

$$\underline{F}^{(i+1)} = \underline{F}^{(i)} + \omega\, \underline{e}^{(i)}$$

No

$|\underline{e}^{(i)}| < \text{Tol. } \epsilon_1$

$|\lambda^{(i)} - \lambda^{(i-1)}| < \text{Tol. } \epsilon_3$

Yes

Convergence

Figure 4.1. Solution of $\mathbb{A}\, \underline{F} = \frac{1}{\lambda}\, \mathbb{B}\, \underline{F}$ Using the Fission Source Power Iteration Method Without Fission Source Renormalization.

would converge to unity. The k_{eff} of the problem would then be simply the product of all of the iteration eigenvalues.[46,47,48]

The matrix inversions required within the iteration scheme were performed directly. Although overrelaxation methods are usually employed only in iterative matrix inversion schemes (or inner iterations),[49] an overrelaxation parameter w, $1 \leq w \leq 2$, is available in the outer iteration in order to hasten the convergence of the solution vector.

The power method is very appealing to neutron diffusion flux calculations because it converges to the largest or fundamental eigenvalue $|\lambda_o| > |\lambda_i|$, $i \neq 0$, and the corresponding eigenvector $\underline{F}_o$ of the given matrix problem. The convergence rate is governed by the dominance ratio, defined as $\max_{i \neq 0} |\lambda_i/\lambda_o|$, in such a way that smaller ratios result in faster convergence. Although the power method will always converge when λ_o is positive and unique, specific matrix properties of $\mathbb{A}$ and $\mathbb{B}$ are sufficient but not always necessary to insure convergence to a positive k_{eff} and everywhere positive neutron flux approximation.[50]

In many problems the order of $\mathbb{A}$ may be quite large, and solution methods which require the direct inversion of $\mathbb{A}$ may not be practical. For the purposes of this report, as in most multigroup calculational schemes, neutron up-scattering will not be permitted. The inversion of $\mathbb{A}$ is then performed by successive group-iteration techniques.

The equations given in Eq. 4.1 have been defined as ordered first by spatial indexing followed by group indexing within each spatial index. It is convenient to reorder these equations so that they are ordered first by group indexing followed by spatial indexing within each group.

After reordering, Eq. 4.1 can be written as

$$(\mathbb{L} + \mathbb{M})\,\underline{F} = \mathbb{T}\,\underline{F} + \frac{1}{\lambda}\,\mathbb{B}\,\underline{F} \tag{4.2a}$$

where

$$\mathbb{A} = \mathbb{L} + \mathbb{M} - \mathbb{T} \tag{4.2b}$$

and: $\mathbb{L}$, the stiffness matrix, results from leakage; $\mathbb{M}$, the mass matrix, results from absorption; $\mathbb{T}$ is the group-to-group scattering transfer matrix; and $\mathbb{B}$ is the fission source production matrix. Assuming K spatial unknowns in each of the G groups, $\mathbb{L}$ and $\mathbb{M}$ are $G \times K$ block diagonal matrices composed of G $K \times K$ matrices $\mathbb{L}_g$ and $\mathbb{M}_g$ of the form

$$\mathbb{L} = \mathrm{Diag}[\,\mathbb{L}_1, \ldots, \mathbb{L}_G\,] \tag{4.3a}$$

$$\mathbb{M} = \mathrm{Diag}[\,\mathbb{M}_1, \ldots, \mathbb{M}_G\,] \tag{4.3b}$$

and $\mathbb{T}$ and $\mathbb{B}$ are in general full block matrices composed of G^2 $K \times K$ matrices $\mathbb{T}_{gg'}$ and $\mathbb{B}_{gg'}$, respectively. Since only downscattering is permitted, $\mathbb{T}$ becomes lower block triangular; $\mathbb{T}_{gg'} = 0$ whenever $g' \geq g$. The matrix inversion, $\underline{\tilde{F}}^{(i+1)} = \mathbb{A}^{-1}\underline{S}^{(i)}$, can then be solved for the GK unknowns

$$\underline{\tilde{F}}^{(i+1)} = \mathrm{Col}\left[\underline{\tilde{F}}_1^{(i+1)} \ \ldots \ \underline{\tilde{F}}_G^{(i+1)}\right] \tag{4.4}$$

by solving successively the following system of group equations:

$$\begin{array}{l}
\text{Do for } g = 1 \text{ to } G: \\
\qquad \underline{S}_g^{(i)} = \displaystyle\sum_{g'=1}^{G} (\mathbb{T}_{gg'} + \mathbb{B}_{gg'})\,\underline{F}_{g'}^{(k)} \\
\qquad\qquad \text{where: } k = \begin{cases} i+1; & g' < g \\ i\ ; & g' \geq g \end{cases} \\
\qquad \underline{\tilde{F}}_g^{(i+1)} = (\mathbb{L}_g + \mathbb{M}_g)^{-1}\,\underline{S}_g^{(i)}
\end{array} \tag{4.5}$$

where the updating of the group fission source by the iteration index $k = i+1$ for $g' < g$ generally enables a faster rate of convergence of the outer iteration than $k = i$.

The desirable convergence properties of a positive eigenvalue and everywhere positive flux solution when using the above group iteration method depend upon the properties of the $K \times K$ spatial matrices for each group g: $\mathbb{L}_g$; $\mathbb{M}_g$; and $\mathbb{T}_{gg'}$ and $\mathbb{B}_{gg'}$, for $g' = 1$ to G. Using the Perron-Frohenius theorem,[50] it can be shown that if $\mathbb{T}_{gg'}$ and $\mathbb{B}_{gg'}$ are all nonnegative for each group g and $\mathbb{L}_g$ and $\mathbb{M}_g$ are both Stieltjes or S-type matrices† for each group g, then the power method will converge to a positive eigenvalue, $\lambda_o > 0$, and a corresponding positive eigenvector, $\underline{F}_o > 0$. These matrix properties naturally depend upon the form of the spatial approximations employed and generally differ for different approximation schemes.

The conventional finite difference approximation has become popular because the spatial matrices which arise from its use exhibit these desirable properties regardless of the size of the mesh regions chosen. The spatial matrices resulting from the linear finite element method, however, are known to exhibit these properties only if the mesh size is restricted by

$$h_k \leq \max_{g=1 \text{ to } G} \{\sqrt{6}\, \ell_{g,k}\} \tag{4.6}$$

where $\ell_{g,k}$ is the diffusion length, $\ell^2_{g,k} = D_{g,k}/\Sigma_{g,k}$, for group g in mesh region k. The spatial matrices resulting from the cubic Hermite

---

†A Stieltjes matrix is a real, irreducible, positive definite matrix with nonpositive off-diagonal elements.[7,50]

finite element method do not exhibit these "desirable" characteristics. Since the eigenvector $\underline{F}$ contains current as well as flux unknowns, convergence to an all-positive solution vector is not desirable.

The properties of the spatial matrices for each group g resulting from the proposed approximation methods can be found by generalizing the proposed trial function forms in each group as

$$U_g(z) = \sum_{k=1}^{K} \left[\underline{P}^{-T}_{g,k}(x)\underline{F}_{g,k} + \underline{P}^{+T}_{g,k}\underline{F}_{g,k+1}\right] \tag{4.7a}$$

$$U^*_g(z) = \sum_{k=1}^{K} \left[\underline{P}^{-*T}_{g,k}(x)\underline{F}^*_{g,k} + \underline{P}^{+*T}_{g,k}\underline{F}^*_{g,k+1}\right] \tag{4.7b}$$

where the $\underline{F}_{g,k}$ are in general column vectors of length N given by:

$$\underline{F}_{g,k} = F_{g,k} \quad (N = 1) \tag{4.8a}$$

for the linear basis function approximations, and

$$\underline{F}_{g,k} = \mathrm{Col}[F_{g,k}, G_{g,k}] \quad (N = 2) \tag{4.8b}$$

for the cubic Hermite basis function approximations. Similar definitions hold for the $\underline{F}^*_{g,k}$. The $\underline{P}^{\pm}_{g,k}(x)$ are column vectors of length N whose elements are functions of z (or x) defined as nonzero only within region k which provide the basis for the approximations. The definitions of the $\underline{P}^{\pm}_{g,k}(x)$ for the proposed approximations are given as follows:

N = 1; Linear Basis Functions:

$$\underline{P}^-_{g,k}(x) = (1-x)\psi^{-1}_{g,k}(0)\psi_{g,k}(x) \tag{4.9a}$$

$$\underline{P}^+_{g,k}(x) = x\,\psi^{-1}_{g,k}(1)\psi_{g,k}(x) \tag{4.9b}$$

N = 2; Cubic Hermite Basis Functions:

$$\underline{P}^{-}_{g,k}(x) = \mathrm{Col}[p_1(x)\psi^{-1}_{g,k}(0)\psi_{g,k}(x)\ ,\ h_k\,\theta p_3(x)D^{-1}_{g,k}(0)\psi^{-1}_{g,k}(0)\psi_{g,k}(x)] \tag{4.10a}$$

$$\underline{P}^{+}_{g,k}(x) = \mathrm{Col}[p_2(x)\psi^{-1}_{g,k}(1)\psi_{g,k}(x)\ ,\ h_k\,\theta p_4(x)D^{-1}_{g,k}(1)\psi^{-1}_{g,k}(1)\psi_{g,k}(x)] \tag{4.10b}$$

where $\psi_{g,k}(x)$ and $D_{g,k}(x)$ are the detailed flux solutions and diffusion coefficients of group g, the polynomials $p_1(x)$ through $p_4(x)$ are defined in Eqs. 3.11, and $0 \leq x \leq 1$ within each region k. Similar definitions hold for the $\underline{P}^{\pm *}_{g,k}(x)$.

Equations 4.7 can be written in matrix form as

$$U_g(z) = \mathbb{P}_g(x)\underline{F}_g \tag{4.11a}$$

$$U^*_g(z) = \mathbb{P}^*_g(x)\underline{F}^*_g \tag{4.11b}$$

where:

$$\underline{F}_g = \mathrm{Col}(F_{g,1}, \ldots, F_{g,K+1}) \tag{4.12a}$$

and $\mathbb{P}_g(x)$ is the K by N(K+1) matrix defined by

$$\mathbb{P}_g(x) = \begin{bmatrix} \underline{P}^{-T}_{g,1}(x) & P^{+T}_{g,1}(x) & & 0 \\ & \ddots & \ddots & \\ 0 & & P^{-T}_{g,K}(x) & P^{+T}_{g,K}(x) \end{bmatrix} \tag{4.12b}$$

$\mathbb{P}^*_g(x)$ is defined similarly. Insertion of these trial function forms into variation equation 2.12b for each group g results in

$$\delta \underline{F}_g^{*T} \int_K \left\{ \dot{\mathbb{P}}_g^{*T} \mathbb{D}_g \dot{\mathbb{P}}_g \underline{F}_g + \mathbb{P}_g^{*T} \sum_{g'=1}^{G} \boldsymbol{\Lambda}_{gg'} \mathbb{P}_{g'} \underline{F}_{g'} \right\} dz = 0 \tag{4.13}$$

where $\mathbb{D}_g$ and $\boldsymbol{\Lambda}_{gg'}$ are K×K diagonal matrices of the form

$$\mathbb{D}_g = \mathbb{D}_g(x) = \mathrm{Diag}\left[D_{g,1}(x), \ldots, D_{g,K}(x) \right] \tag{4.14a}$$

and

$$\boldsymbol{\Lambda}_{gg'} = \boldsymbol{\Lambda}_{gg'}(x) = \mathrm{Diag}\left[\Lambda_{gg',1}(x), \ldots, \Lambda_{gg',K}(x) \right] \tag{4.14b}$$

The quantity $\dot{\mathbb{P}}_g$ represents the derivative of $\mathbb{P}_g(z)$ with respect to z, and the integration over K denotes integration over the entire range of z; $z_1 \leqslant z \leqslant z_{K+1}$.

$D_{g,k}$ and $\Lambda_{gg',k}$,

$$\Lambda_{gg',k} = \left(\Sigma_{tg} - \Sigma_{gg'} - \frac{1}{\lambda} \chi_g \nu \Sigma_{fg'} \right)_{\text{in region } k} \tag{4.14c}$$

are the group material constants in mesh region k, and are usually dependent on x. $\boldsymbol{\Lambda}_{gg'}$ can thus be conveniently expanded as

$$\boldsymbol{\Lambda}_{gg'} = \boldsymbol{\Lambda}_g^A - \boldsymbol{\Lambda}_{gg'}^S - \frac{1}{\lambda} \boldsymbol{\Lambda}_{gg'}^F \tag{4.14d}$$

Allowing arbitrary variations in each element of $\underline{F}_g^*$ for each group g in Eq. 4.13 results in the matrix equations

$$(\mathbb{L}_g + \mathbb{M}_g)\underline{F}_g = \sum_{g'=1}^{G} \left(\mathbb{T}_{gg'} + \frac{1}{\lambda} \mathbb{B}_{gg'} \right) \underline{F}_{g'}; \quad g = 1 \text{ to } G \tag{4.15}$$

as described in Eqs. 4.1 through 4.5, where:

$$\mathbb{L}_g = \int_K \dot{\mathbb{P}}_g^{*T} \mathbb{D}_g \dot{\mathbb{P}}_g \, dz \tag{4.16a}$$

$$\mathbb{M}_g = \int_K \mathbb{P}_g^{*T} \boldsymbol{\Lambda}_g^A \mathbb{P}_g \, dz \tag{4.16b}$$

$$\mathbb{T}_{gg'} = \int_K \mathbb{P}_g^{*T} \boldsymbol{\Lambda}^S_{gg'} \mathbb{P}_{g'}\, dz \tag{4.16c}$$

$$\mathbb{B}_{gg'} = \int_K \mathbb{P}_g^{*T} \boldsymbol{\Lambda}^F_{gg'} \mathbb{P}_{g'}\, dz \tag{4.16d}$$

These matrices are N(K+1) by N(K+1) block tridiagonal of similar form whose N×N submatrices are integrals of N×N dyads. The k^{th} row of the product $\boldsymbol{\Lambda}_{gg'} \underline{F}_{g'}$ is, for example:

$$\begin{aligned}
[\boldsymbol{\Lambda}_{gg'} \underline{F}_{g'}]_k = & \left\{ \int_0^1 h_{k-1} \Lambda_{gg',k-1}(x) \underline{P}^{+*}_{g,k-1}(x) \underline{P}^{-T}_{g',k-1}(x)\, dx \right\} F_{g',k-1} \\
& + \left\{ \int_0^1 h_{k-1} \Lambda_{gg',k-1}(x) \underline{P}^{+*}_{g,k-1}(x) \underline{P}^{+T}_{g',k-1}(x)\, dx \right. \\
& \left. + \int_0^1 h_k \Lambda_{gg',k}(x) \underline{P}^{-*}_{g,k}(x) \underline{P}^{-T}_{g',k}(x)\, dx \right\} F_{g',k} \\
& + \left\{ \int_0^1 h_k \Lambda_{gg',k}(x) \underline{P}^{-*}_{g,k}(x) \underline{P}^{+T}_{g',k}(x)\, dx \right\} F_{g',k+1}
\end{aligned} \tag{4.17}$$

These matrix relationships allow presentation of the following matrix properties.

<u>Theorem 1</u>: $\mathbb{L}_g$ and $\mathbb{M}_g$ are guaranteed to be positive definite whenever the detailed weighting functions $\psi^*_{g,k}(z)$ have a similar shape to that of the detailed flux solutions $\psi_{g,k}(z)$, as given by:

$$\psi^*_{g,k}(z) = C_{g,k}\psi_{g,k}(z) \tag{4.18}$$

where $C_{g,k}$ is a positive constant for each energy group g and each region k.

Proof: Under these conditions, $\underline{P}^{*\pm}_{g,k} = C_{g,k}\underline{P}^{\pm}_{g,k}$;

hence,

$$\mathbb{P}^*_g = \mathbb{C}_g \mathbb{P}_g \tag{4.19a}$$

where

$$\mathbb{C}_g = \text{Diag}\,(C_{g,1}, \ldots, C_{g,K}) \tag{4.19b}$$

First consider $\mathbb{M}_g$. Given any arbitrary constant nonzero vector $\underline{q}$,

$$\underline{q}^T \mathbb{M}_g \underline{q} = \int_K (\mathbb{P}_g \underline{q})^T \mathbb{C}^T_g \boldsymbol{\Lambda}^A_g \mathbb{P}_g \underline{q}\, dz \tag{4.20}$$

and since $\mathbb{C}_g$ and $\boldsymbol{\Lambda}^A_g$ are both positive block diagonal matrices with diagonal submatrices, their product can be factored into

$$\mathbb{C}^T_g \Lambda^A_g = \left[(\boldsymbol{\Lambda}^A_g \mathbb{C}_g)^{\frac{1}{2}} \right]^T (\mathbb{C}_g \boldsymbol{\Lambda}^A_g)^{\frac{1}{2}} \tag{4.21}$$

Therefore,

$$\underline{q}^T \mathbb{M}_g \underline{q} = \int_K \left[(\mathbb{C}_g \boldsymbol{\Lambda}^A_g)^{\frac{1}{2}} \mathbb{P}_g \underline{q} \right]^T \left[(\mathbb{C}_g \boldsymbol{\Lambda}^A_g)^{\frac{1}{2}} \mathbb{P}_g \underline{q} \right] dz \tag{4.22a}$$

$$= \int_K \mathbb{R}^T \mathbb{R}\, dz \tag{4.22b}$$

which is always greater than zero for arbitrary nonzero $\underline{q}$. Hence, by definition, $\mathbb{M}_g$ is positive definite. A similar proof holds for $\mathbb{L}_g$ using Eq. 4.16a.

The following corollaries immediately result.

Corollary 1: If Rayleigh-Ritz Galerkin weighting, $\underline{U}^* = \underline{U}$, is used in the approximation, then $\mathbb{L}_g$ and $\mathbb{M}_g$ are positive definite.

Corollary 2: $\mathbb{L}_g$ and $\mathbb{M}_g$ resulting from the finite element methods using linear and cubic Hermite basis functions are both positive definite.

Corollary 3: If $\mathbb{L}_g$ and $\mathbb{M}_g$ are positive definite, then so is the matrix $(\mathbb{L}_g + \mathbb{M}_g)$. These three matrices are then also symmetric.

It is also interesting to note the properties of $\mathbb{L}_g$ and $\mathbb{M}_g$ for cases of symmetry; that is, when the material properties and detailed flux solutions are symmetric about the center of each coarse mesh region k. Such symmetry occurs in regular repeating reactor geometries, and is denoted by:

$$D_{g,k}(x) = D_{g,k}(1-x) \tag{4.23a}$$

and

$$\Lambda_{g,k}(x) = \Lambda_{g,k}(1-x) \tag{4.23b}$$

Hence,

$$\psi_{g,k}(x) = \psi_{g,k}(1-x) \tag{4.23c}$$

and

$$\eta_{g,k}(x) = -\eta_{g,k}(1-x) \tag{4.23d}$$

and similarly for the weighting fluxes and currents. Under such conditions, the $\underline{P}^{+}_{g,k}(x)$ and $\underline{P}^{-}_{g,k}(x)$ support functions can be found by inspection of Eqs. 4.9 and 4.10 to obey the following symmetries:

For N = 1:

$$\underline{P}^{\mp}_{g,k}(x) = \underline{P}^{\pm}_{g,k}(1-x) \tag{4.24a}$$

$$\dot{\underline{P}}^{\mp}_{g,k}(x) = -\dot{\underline{P}}^{\pm}_{g,k}(1-x) \tag{4.24b}$$

For N = 2:

$$\underline{P}^{\mp}_{g,k}(x) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \underline{P}^{\pm}_{g,k}(1-x) \qquad (4.24c)$$

$$\dot{\underline{P}}^{\mp}_{g,k}(x) = -\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \dot{\underline{P}}^{\pm}_{g,k}(1-x) \qquad (4.24d)$$

where the following symmetries of the polynomials defined in Eqs. 3.11 have been used:

$$\begin{aligned} p_1(x) &= p_2(1-x) \\ p_3(x) &= -p_4(1-x) \\ q_1(x) &= -q_2(1-x) \\ q_3(x) &= q_4(1-x) \end{aligned} \qquad (4.25)$$

Similar identities with identical signs hold for the weighting quantities $\underline{P}^{\pm *}_{g,k}(x)$.

Theorem 2: For cases of symmetry, as given above, the matrices $\mathbb{L}_g$, $\mathbb{M}_g$, $\mathbb{T}_{gg'}$, and $\mathbb{B}_{gg'}$ are all symmetric regardless of the relation of $\psi^*_{g,k}(x)$ to $\psi_{g,k}(x)$.

Proof: Referring to Eq. 4.17, $\mathbb{L}_g$, for example, is symmetric only if

$$\int_0^1 h_k D_{g,k}(x) \dot{\underline{P}}^{+*}_{g,k}(x) \dot{\underline{P}}^{-T}_{g,k}(x)\, dx = \int_0^1 h_k D_{g,k}(x) \dot{\underline{P}}^{-*}_{g,k}(x) \dot{\underline{P}}^{+T}_{g,k}(x)\, dx \qquad (4.26)$$

This can be shown for any N by changing variables in one of the integrals from x to 1-x′ and using the symmetry properties of Eqs. 4.23 and 4.24. Similar proofs hold for the other matrices.

It is unfortunate that the above symmetry conditions do not allow direct proof that $\mathbb{L}_g$ and $\mathbb{M}_g$ have positive diagonal elements and are also diagonally dominant (for at least one row) for arbitrary positive and symmetric detailed flux solutions. Under such conditions, $\mathbb{L}_g$ and $\mathbb{M}_g$ would then be positive definite, since they are block tridiagonal with nonzero diagonal elements and hence irreducible. Instead, these conditions can be used to obtain a set of algebraic equations which, for completely arbitrary detailed flux solutions, must be satisfied in order that $\mathbb{L}_g$ and $\mathbb{M}_g$ be positive definite.

The requirement that $\mathbb{L}_g$ and $\mathbb{M}_g$ be positive definite is useful only in the inversion of $(\mathbb{L}_g+\mathbb{M}_g)$. Although the inversion can always be performed using Gaussian elimination techniques, the property of positive definiteness allows the use of Cholesky's method, discussed in the next section, which is faster and requires less computer storage.

4.2 Calculational and Programming Techniques

Calculation of the one-dimensional subassembly detailed fluxes, currents, and adjoint solutions, as well as detailed "exact" or reference solutions, were performed using the program REF2G described in section 1 of Appendix D. Assuming subassembly k to be divided into N homogeneous intervals at nodes t_i and of width hs_i, the program uses fine mesh linear finite element approximations to calculate the detailed flux solutions for each group. Omitting group subscripts, the detailed flux solution for each group in subassembly k is represented by a set of N+1 points

$$\psi_k(x) = \{\psi_{k,i} : i = 1, N+1\} \tag{4.27}$$

where $\psi_k(x)$ is linear between points. Detailed group current solutions $\eta_k(x)$ are represented by a set of N points

$$\eta_k(x) = \{\eta_{k,i} : i = 1, N\} \tag{4.28}$$

which are found from the converged flux solutions by Fick's law

$$\eta_{k,i} = \frac{1}{hs_i} D_{k,i} [\psi_{k,i} - \psi_{k,i+1}]; \quad i = 1, N \tag{4.29}$$

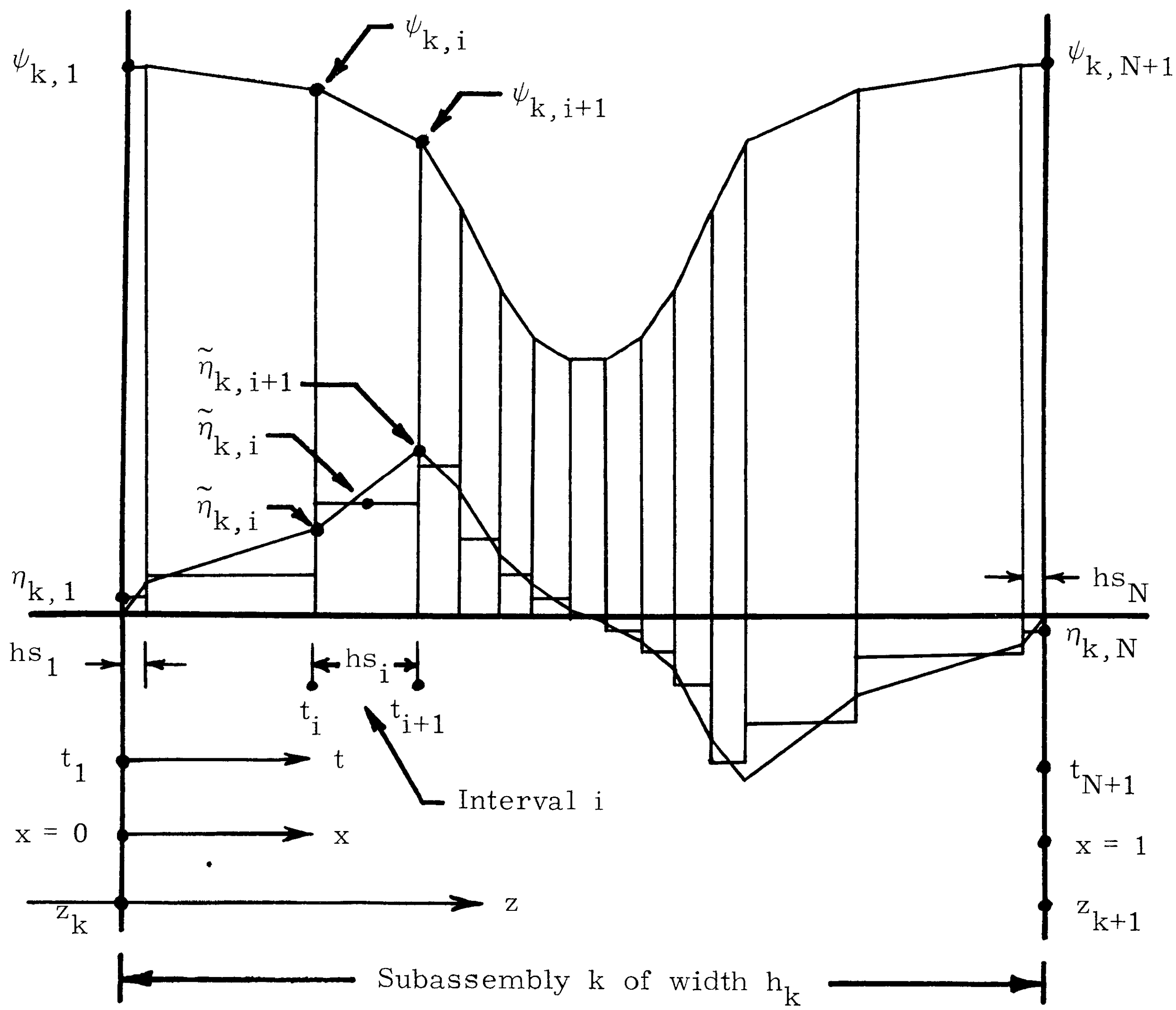
where $D_{k,i}$ is the diffusion constant homogeneous in interval i. $\eta_k(x)$ is of constant value, $\eta_{k,i}$, within each interval. The forms of these solutions are illustrated in Figure 4.2.

In order to approximate the symmetry boundary conditions imposed on the detailed subassembly flux solutions, small intervals hs_1 and hs_N are defined at the edges of each subassembly. The detailed current solutions can then be made to have zero boundary values by setting $\eta_{k,1}$ and $\eta_{k,N}$ to zero. However, since the currents in each interval are defined as inversely proportional to the mesh size, the calculated boundary currents using this scheme may not be small enough to be negligible.

Explicit zero current boundary conditions can be imposed on the detailed current solutions by transforming the above discontinuous current $\eta_k(x)$ into a continuous current solution $\tilde{\eta}_k(x)$ represented by a set of N+1 points

$$\tilde{\eta}_k(x) = \{\tilde{\eta}_{k,i} : i = 1 \text{ to } N+1\} \tag{4.30}$$

where $\tilde{\eta}_k(x)$ is linear between points, as also illustrated in Figure 4.2. By seeking to minimize the mean square error between $\eta_k(x)$ and $\tilde{\eta}_k(x)$ within each interval i, variational techniques yield the following set of N-1 equations for each group:



$$x = \frac{1}{h_k}(z - z_k)$$

$0 \leq x \leq 1$; for each subassembly k; k = 1 to K.

$$y = \frac{1}{hs_i}(t - t_i)$$

$0 \leq y \leq 1$; for each interval i in subassembly k; i = 1 to N.

Figure 4.2. Subassembly Notations and Detailed Solutions

$$\left(\frac{1}{6}\,hs_{i-1}\right)\tilde{\eta}_{k,i-1} + \frac{1}{3}\left(hs_{i-1} + hs_i\right)\tilde{\eta}_{k,i} + \left(\frac{1}{6}\,hs_i\right)\tilde{\eta}_{k,i+1}$$

$$= \left(\frac{1}{2}\,hs_{i-1}\right)\eta_{k,i-1} + \left(\frac{1}{2}\,hs_i\right)\eta_{k,i}\ ; \quad i = 2 \text{ to } N \tag{4.31}$$

These equations, given the $\eta_{k,i}$ from Eq. 4.29, are easily solved for the $\tilde{\eta}_{k,i}$, i = 2 to N, where $\tilde{\eta}_{k,1}$ and $\tilde{\eta}_{k,N+1}$ are set to zero. Both forms of the detailed currents, $\eta_k(x)$ and $\tilde{\eta}_k(x)$, are allowed for use in the proposed approximation methods.

The proposed methods using linear and cubic Hermite basis functions have been programmed into computer codes LINEAR and CUBIC which are described respectively in sections 2 and 3 of Appendix D.

The matrix elements required for use in the approximation methods are integrals of products of subassembly detailed solutions and polynomial functions. These integrals are calculated, for each index k, from the basic integral unit

$$BIU_k = \int_0^1 f_k(x) g_k(x) C_k(x)\, x^n\, h_k\, dx \tag{4.32}$$

where the functions $f_k(x)$ and $g_k(x)$ represent flux and/or current solutions for same or different groups. These functions may be either constant within each interval

$$f_k(x) = \left\{ f_{k,i} : i = 1 \text{ to } N \right\} \tag{4.33}$$

or of linear form within each interval

$$f_k(x) = \left\{ (1-y) f_{k,i} + y\, f_{k,i+1} : i = 1 \text{ to } N \right\} \tag{4.34}$$

where $y = \frac{1}{hs_i}\,(t - t_i)$, as defined in Figure 4.2. $C_k(x)$ represents a group nuclear constant which is homogeneous in each interval

$$C_k(x) = \{ C_{k,i} : i = 1 \text{ to } N \} \tag{4.35}$$

and n is a positive integer exponent in the range $0 \leq n \leq 6$. Since the following remarks concern only subassembly k, the index k is dropped for simplicity.

The basic integral unit can be broken into integrals over each interval by transforming variables from x to y. The result

$$BIU_k = \frac{1}{h_k^n} \sum_{i=1}^{N} hs_i \int_0^1 f_i(y) g_i(y) C_i (t_i + hs_i y)^n \, dy \tag{4.36}$$

can be integrated analytically by expanding $(t_i + hs_i y)^n$ into a binomial series. The results of these integrations for any n depend only on the given forms of $f(x)$ and $g(x)$, and are summarized in Table 4.1.

The coarse mesh flux-weighting homogenization calculations were performed using the above basic integral unit with $n = 0$. In these calculations a linear form of $f(x)$, representing the detailed subassembly flux solutions from REF2G, and a constant value of $g(x) = 1$ were used.

Once the elements of the matrices of the approximation methods have been formed, considerable computer storage can be saved by collapsing the sparse band-structured matrices into full matrix form using row index transformations. In this way, a $N \times N$ tridiagonal matrix $\mathbb{L}$ resulting from the use of linear basis functions can be stored as the $N \times 3$ matrix $\mathbb{L}'$ by

$$(\mathbb{L}')_{ik} = (\mathbb{L})_{ij} \tag{4.37}$$

where $k = j + 2 - i$, and k values outside $1 \leq k \leq 3$ are omitted.

Table. 4.1. Calculation of the Basic Integral Unit in Subassembly k.

1. Constant f(x) and constant g(x) in each interval i:

$$BIU_k = \frac{1}{h_k^n} \sum_{i=1}^{N} \left\{ hs_i f_i g_i C_i \sum_{\ell=0}^{n} b_\ell^n \, t_i^{n-\ell} \, hs_i^\ell \, \frac{1}{(\ell+1)} \right\}$$

2. Linear f(x) and constant g(x) in each interval i:

$$BIU_k = \frac{1}{h_k^n} \sum_{i=1}^{N} \left\{ hs_i g_i C_i \sum_{\ell=0}^{n} b_\ell^n t_i^{n-\ell} hs_i^\ell \left[\frac{f_i}{(\ell+1)(\ell+2)} + \frac{f_{i+1}}{(\ell+2)} \right] \right\}$$

3. Linear f(x) and linear g(x) in each interval i:

$$BIU_k = \frac{1}{h_k^n} \sum_{i=1}^{N} \left\{ hs_i C_i \sum_{\ell=0}^{n} b_\ell^n \, t_i^{n-\ell} \, hs_i^\ell \left[\frac{2 f_i g_i}{(\ell+1)(\ell+2)(\ell+3)} + \frac{f_i g_{i+1} + f_{i+1} g_i}{(\ell+2)\,(\ell+3)} + \frac{f_{i+1} g_{i+1}}{(\ell+3)} \right] \right\}$$

where

$$b_\ell^n = \frac{n!}{\ell!\,(n-\ell)!}$$

is the binomial series coefficient.

Similarly, a $N \times N$ matrix $\mathbb{L}$ of half-band width equal to three, which results from the use of cubic Hermite basis functions, can be stored as the $N \times 6$ matrix $\mathbb{L}'$, as given above, where in this case:

$$k = j - i + 3 + \begin{cases} 1 & \text{for } i \text{ odd} \\ 0 & \text{for } i \text{ even} \end{cases} \tag{4.38}$$

and k values outside $1 \leq k \leq 6$ are omitted.

In those cases where the mass and stiffness matrices are both positive definite, Cholesky's method of matrix factorization,

$$\mathbb{L} = \mathbb{G}\,\mathbb{G}^T \tag{4.39}$$

where $\mathbb{L}$ is positive definite and $\mathbb{G}$ is lower triangular, can be used to solve the matrix inversion for each group in the power method. The matrix elements $g_{ij} = (\mathbb{G})_{ij}$ are calculated from the elements $\ell_{ij} = (\mathbb{L})_{ij}$ by the following algorithm:[51]

For each $j = 1$ to N:

$$g_{jj} = \left[\ell_{jj} - \sum_{k=1}^{j-1} g_{jk}^2 \right]^{\frac{1}{2}}$$

For each $i = j+1$ to N:

$$g_{ij} = \left[\ell_{ij} - \sum_{k=1}^{j-1} g_{ik} g_{jk} \right] / g_{jj} \tag{4.40}$$

Similar algorithms of a more complex form are used in the computer codes in conjunction with the matrix collapsing schemes given above.

Chapter 5
NUMERICAL RESULTS

5.1 Nuclear Constants and Subassembly Geometry

The effectiveness and accuracy of the proposed approximation methods were examined using one-dimensional, one- and two-group reactor configurations composed of representative PWR fuel subassemblies. Four separate subassemblies with identical geometry but different number constants are considered. Each subassembly is represented as an 18-cm, homogeneous fuel region of low, medium, or high enrichment, surrounding a 1-cm centrally located absorption rod or water channel. Two-group regional nuclear constants used to represent such PWR subassembly geometries are given in Table 5.1,[52] where all fission neutrons are assumed to be born in the fast group. These constants were collapsed into representative one-group constants using the standard infinite medium group reduction procedure for two groups:

$$\langle \Sigma \rangle_{1G} = \frac{1}{(1+\alpha)} (\Sigma_1 + \alpha \Sigma_2) \tag{5.1}$$

where Σ_1 and Σ_2 are macroscopic cross sections for the fast and thermal groups, respectively, and α is the infinite medium thermal to fast flux ratio.[53] The resulting one-group regional constants for the fuel and rod regions are given in Table 5.2, where the flux ratios of the three fuel regions have been averaged in order to collapse the absorption rod constants.

Table 5.1. Representative Two-Group, 18-cm, PWR Subassembly Regional Nuclear Constants. $\chi_1 = 1.0$; $\chi_2 = 0.0$.

Region Material	Σ_T	$\nu\Sigma_f$	D	Σ_{21}
Fuel A: Low w/o	.0259	.00485	1.396	.0179
	.0532	.0636	.388	
Fuel B: Medium w/o	.0260	.00553	1.397	.0172
	.0710	.102	.389	
Fuel C: High w/o	.0261	.00659	1.399	.0168
	.0832	.129	.387	
Absorption Rod	.0452	0.0	1.0	0.0
	.959	0.0	1.0	
Water	.0383	0.0	1.63	.0380
	.0108	0.0	.275	

Fast group constants appear first for each region material, followed by thermal group constants. Fission neutrons are assumed to be born in the fast group only.

Table 5.2. Representative One-Group, 18-cm, PWR Subassembly Regional Nuclear Constants.

Region Material	Σ_T	$\nu\Sigma_f$	D
Fuel A: Low w/o	.0329	.0199	1.14
Fuel B: Medium w/o	.0348	.0244	1.20
Fuel C: High w/o	.0357	.0272	1.23
Absorption Rod	.235	0.0	1.0
Water	.0136	0.0	.414

Four subassembly configurations, labeled A through D, were used in the one- and two-group test configurations, and are illustrated in Figure 5.1. Subassemblies labeled A, B, and C contain homogeneous fuel of low, medium, and high enrichment, respectively, surrounding the 1-cm absorption rod while subassembly D contains low enriched homogeneous fuel surrounding a 1-cm water channel.

5.2 Subassembly Detailed Solutions and Homogenized Nuclear Constants

The detailed flux and current solutions for each subassembly were found using the computer code REF2G with symmetry boundary conditions and a 68-mesh region per subassembly geometry as indicated in Figure 5.2. The resulting one-group detailed flux solutions for each subassembly are shown in Figure 5.3. The resulting two-group detailed flux and adjoint flux solutions for each subassembly are shown in Figures 5.4 and 5.5, respectively.

Homogeneous subassembly group constants for use in the finite element approximations were found by flux weighting the group cross sections in each subassembly by the corresponding subassembly detailed group flux solutions. The resulting homogenized one-group constants for each subassembly are given in Table 5.3, and the resulting homogenized two-group constants are given in Table 5.4. The results of homogenizing the diffusion coefficient as the transport cross section, $1/\langle 1/D \rangle$, as well as by direct homogenization, $\langle D \rangle$, are included in the tables. The results of both schemes were found to differ at most by only 2%. The directly homogenized diffusion coefficients, $\langle D \rangle$, were used in the finite element approximations.

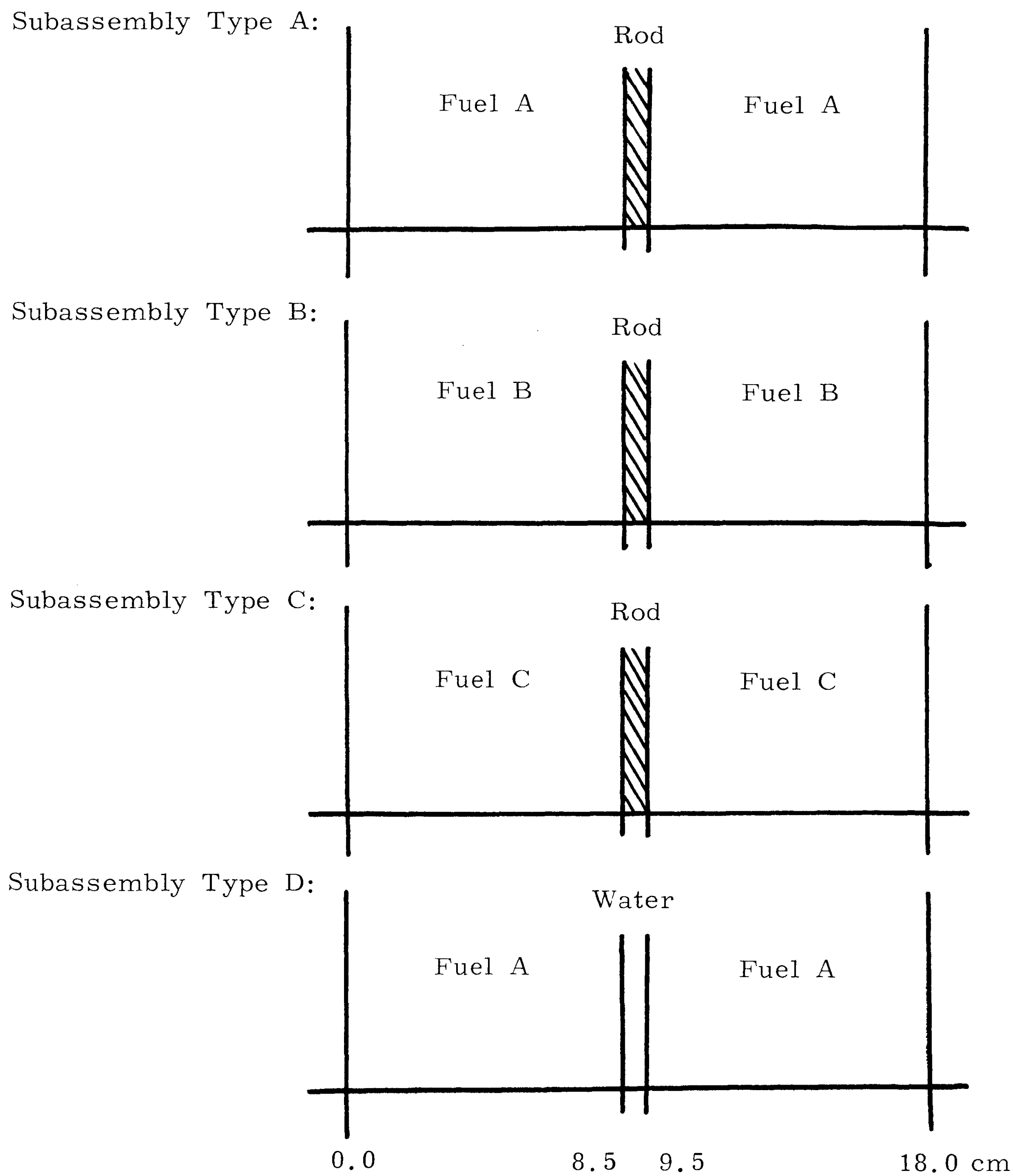
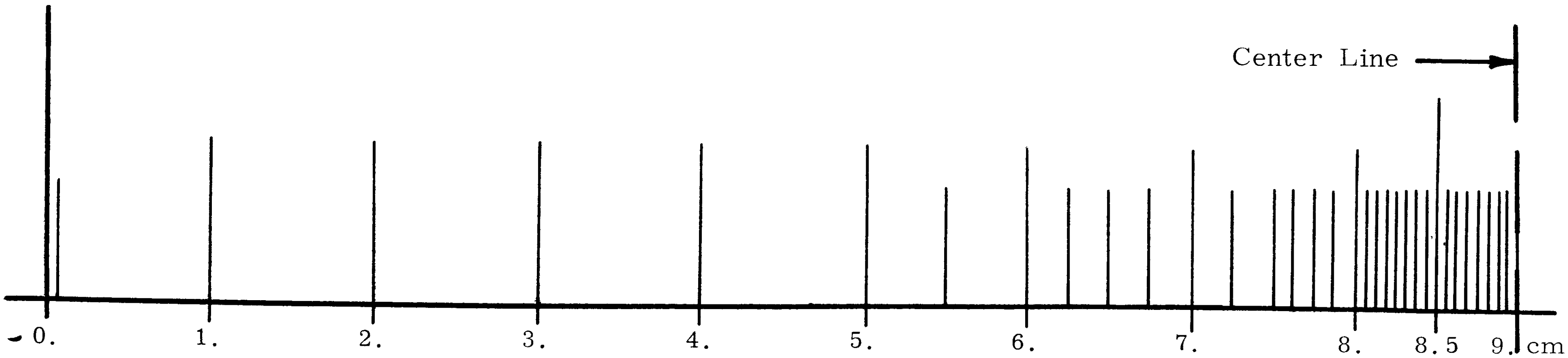


Figure 5.1. Subassembly Configuration Geometries



Symmetric Partitioning:

1 (1/16 cm) + 1 (15/16) + 4 (1) + 2 (1/2) + 6 (1/4) + 4 (1/8) + 8 (1/16) + 8 (1/16)

Figure 5.2. Mesh Geometry in Half a Subassembly.
Detailed flux and current solution calculations use this 68 intervals/subassembly geometry in each subassembly type.

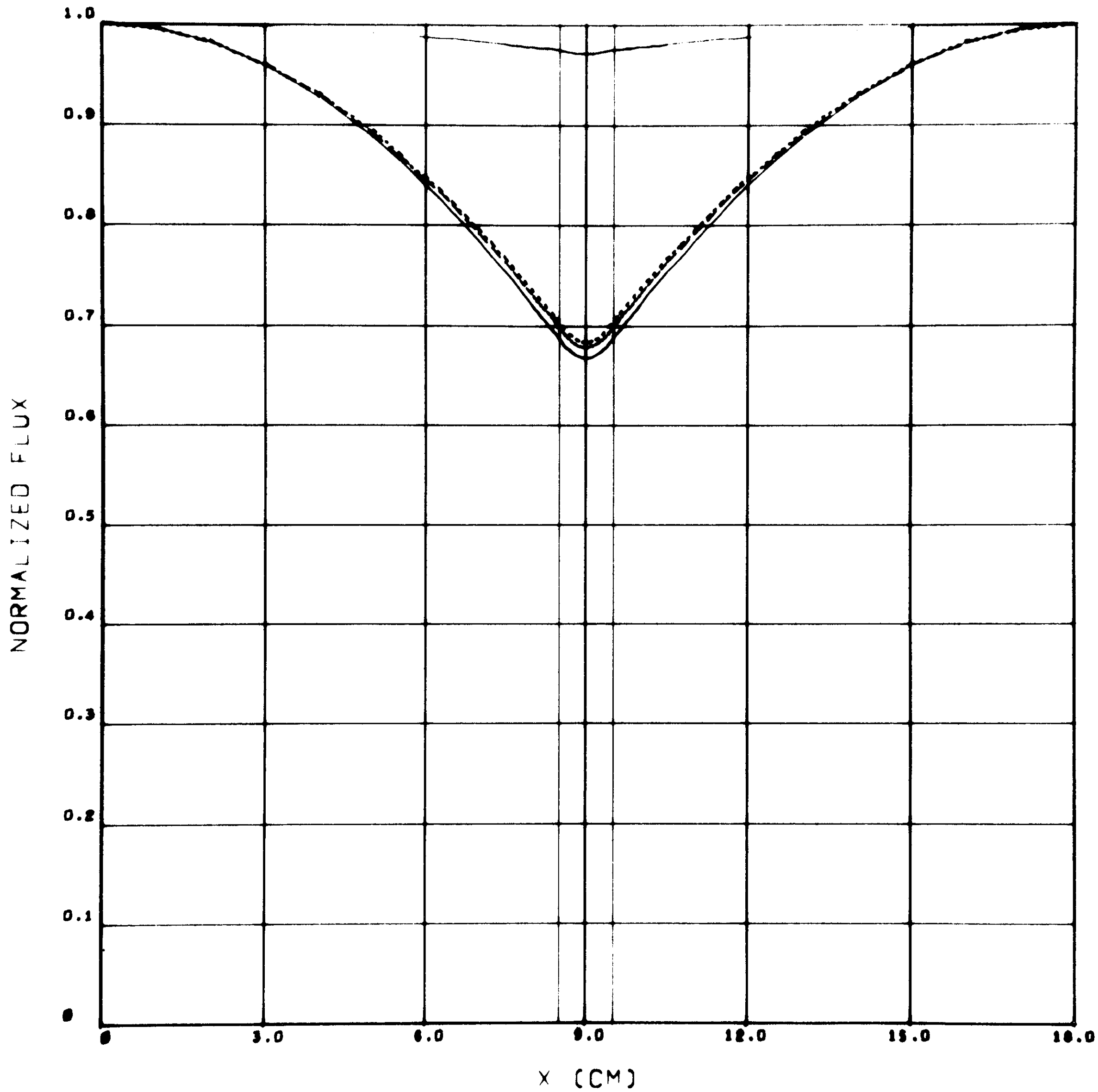


Figure 5.3. Subassembly Detailed Flux Solutions for the One-Group Case

Subassembly Type A ———— (lower curve)

Subassembly Type B - - - - -

Subassembly Type C ....

Subassembly Type D ———— (upper curve)

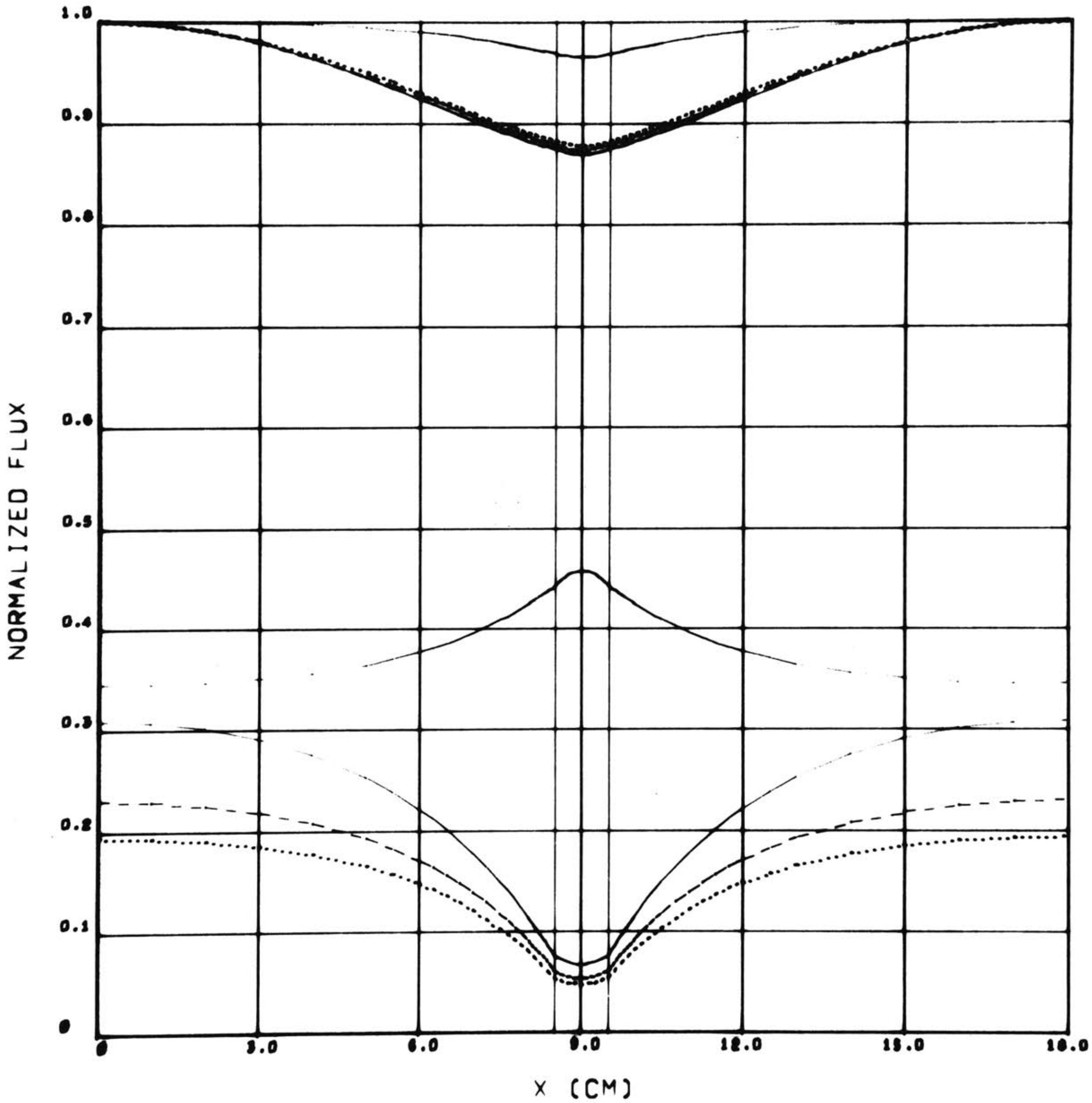


Figure 5.4. Subassembly Detailed Flux Solutions for the Two-Group Case

The fluxes are normalized by fast flux values so that the thermal fluxes appear in the lower portion of the figure.

Subassembly Type A ———— (lower curves)

Subassembly Type B - - - - -

Subassembly Type C ....

Subassembly Type D ———— (upper curves)

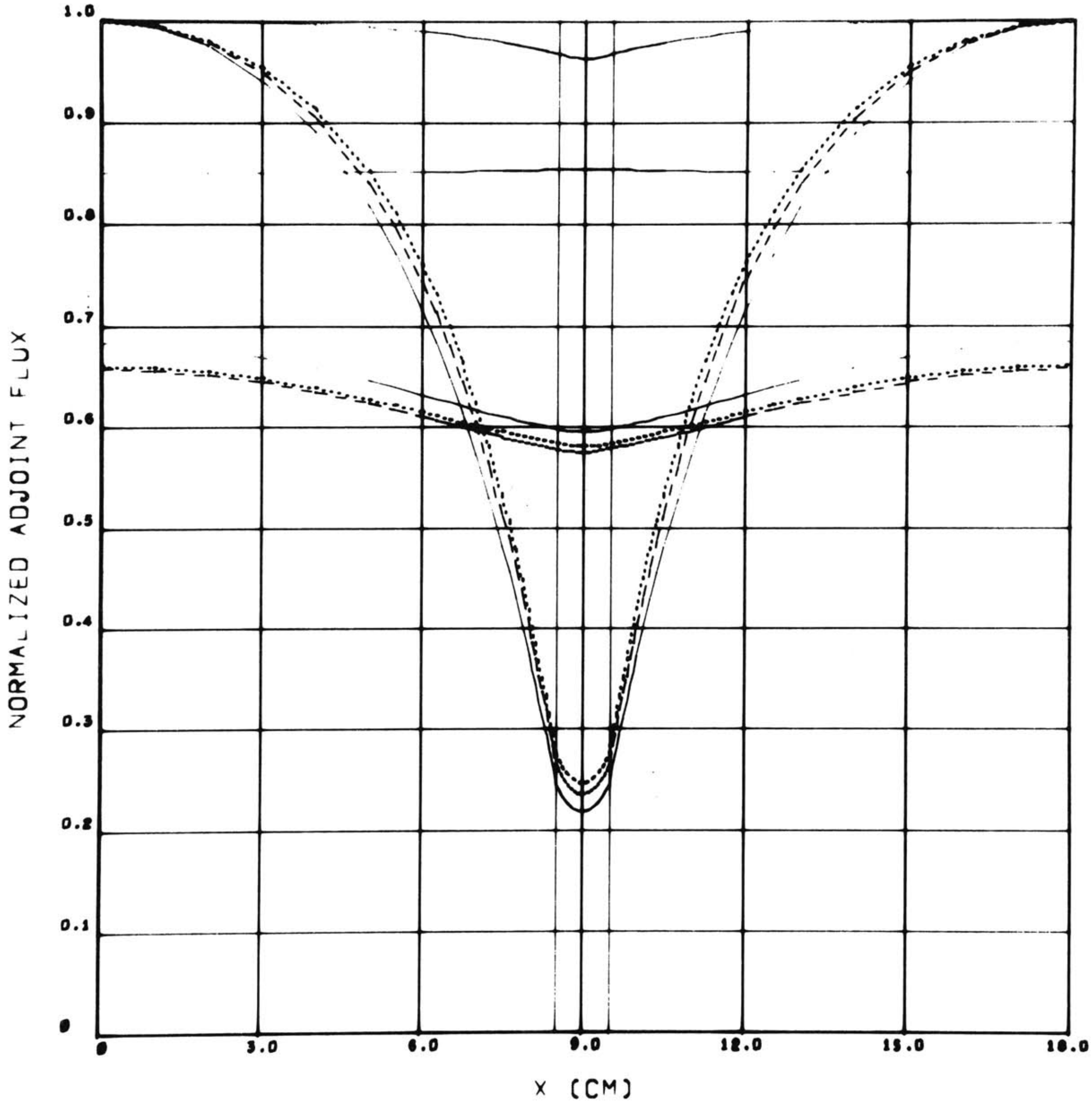


Figure 5.5. Subassembly Detailed Adjoint Flux Solutions for the Two-Group Case

Subassembly Type A ———— (lower curves)

Subassembly Type B - - - - -

Subassembly Type C ....

Subassembly Type D ———— (upper curves)

Table 5.3. Homogenized Subassembly One-Group Nuclear Constants.

Subassembly	$\langle \Sigma_T \rangle$	$\langle \nu\Sigma_f \rangle$	$\langle D \rangle$	$1/\langle 1/D \rangle$
A	.04149392	.01905379	1.134047	1.133253
B	.04341140	.02335046	1.191397	1.189765
C	.04431869	.02602374	1.220054	1.217887
D	.03184731	.01881458	1.100401	1.040479

Table 5.4. Homogenized Subassembly Two-Group Nuclear Constants.
$\chi_1 = 1.0$; $\chi_2 = 0.0$.

Sub-assembly	$\langle \Sigma_T \rangle$	$\langle \nu\Sigma_f \rangle$	$\langle D \rangle$	$\langle \Sigma_{21} \rangle$	$1/\langle 1/D \rangle$
A	.02688787	.004601752	1.379526	.01698379	1.371911
	.06812834	.06255182	.3980863		.3919533
B	.02698495	.005246314	1.379480	.01631765	1.37185
	.08647802	.1002221	.3996499		.3931874
C	.02708207	.006251160	1.379433	.01593619	1.371787
	.09893614	.1266822	.3980142		.391310
D	.02657213	.004587110	1.412467	.0189895	1.410790
	.05034835	.05932253	.3804001		.3775656

Fast group constants appear first for each subassembly, followed by thermal group constants.

Before applying the proposed approximation methods to complex reactor geometries, test runs were performed in order to evaluate the differences between using either flux or adjoint flux weighting, and using current solutions of either constant or linear form in each subassembly interval, as described in section 4.2. The test problem consisted of three consecutive Type A subassemblies with symmetry boundary conditions imposed on each end so that the converged eigenvalue λ (k_{eff}) should be identical to that of the detailed flux solution of subassembly A. Entire subassemblies were chosen as the mesh regions so that the proposed synthesis methods should converge to flux values of unity, and current values of zero. The numerical results of these tests for the one- and two-group cases are summarized in Table 5.5. Although the choice of weighting function did not influence the results for either approximation, use of current solutions of the linear form enables better eigenvalue accuracy. In addition, the results when using currents of linear form converged to flux values of unity and current values of zero, as expected, while results using the constant current form produced errors of about 0.5% in the converged flux and 0.01% in the converged current at interior points. Although the difference in accuracy between the use of these different current forms is small, the small flux and current errors resulting from the use of the constant current form may lead to larger errors in larger and more complex problems. For the above reasons, the linear current form was used in the following case studies. Adjoint weighting was also used. Although the use of adjoint weighting has not been shown to guarantee the success of Cholesky's method in the numerical solution scheme, no difficulties with its use were ever encountered.

Table 5.5. Test Results Using Three Consecutive Type A Subassemblies.
$\% \lambda = [(\lambda_{Sub.\ A} - \lambda_{Conv.})/\lambda_{Sub.\ A}] \times 100\%$.

Synthesis Approximation	Weighting Function	Form of Currents	Converged λ	% λ
		ONE GROUP: $\lambda_{Sub.\ A} = 0.459194$		
Linear	FLUX	Constant	.459363	-.036%
Linear	FLUX	Linear	.459254	-.013%
Cubic	FLUX	Constant	.459363	-.036%
Cubic	FLUX	Linear	.459254	-.013%
		TWO GROUPS: $\lambda_{Sub.\ A} = 0.751095$		
Linear	FLUX	Linear	.751284	-.025%
Linear	ADJOINT	Constant	.7513818	-.038%
Linear	ADJOINT	Linear	.751284	-.025%
Cubic	FLUX	Linear	.751284	-.025%
Cubic	ADJOINT	Constant	.7513818	-.038%
Cubic	ADJOINT	Linear	.751284	-.025%

5.3 Case Studies and Results

Four one-dimensional reactor configurations, each made up of different combinations of types of subassemblies, are considered in the case studies below. One-group calculations were performed only for the first case, and two-group calculations were performed in all cases. Entire 18-cm subassemblies were used as coarse mesh regions in each case, while the effect of using half-subassembly mesh regions was also included in Case 1. The geometry and subassembly configurations of the case studies are shown in Figure 5.6.

Three separate approximation methods were used to calculate converged detailed flux solutions for comparison in each case. They are:

1. The proposed approximation methods using heterogeneous nuclear constants and subassembly detailed flux solutions for coarse mesh solutions.
2. The finite element methods using subassembly homogenized nuclear constants for coarse mesh solutions.
3. The linear finite element method for fine mesh reference solutions.

Calculations of both the proposed approximation and the coarse mesh finite element method using linear basis functions were performed using program LINEAR, while the corresponding cubic Hermite basis function approximations were performed using program CUBIC. The fine mesh reference solutions were calculated using program REF2G, and the results of these approximations were compared and analyzed by program ANALYZE. Descriptions of these programs are given in Appendix D.

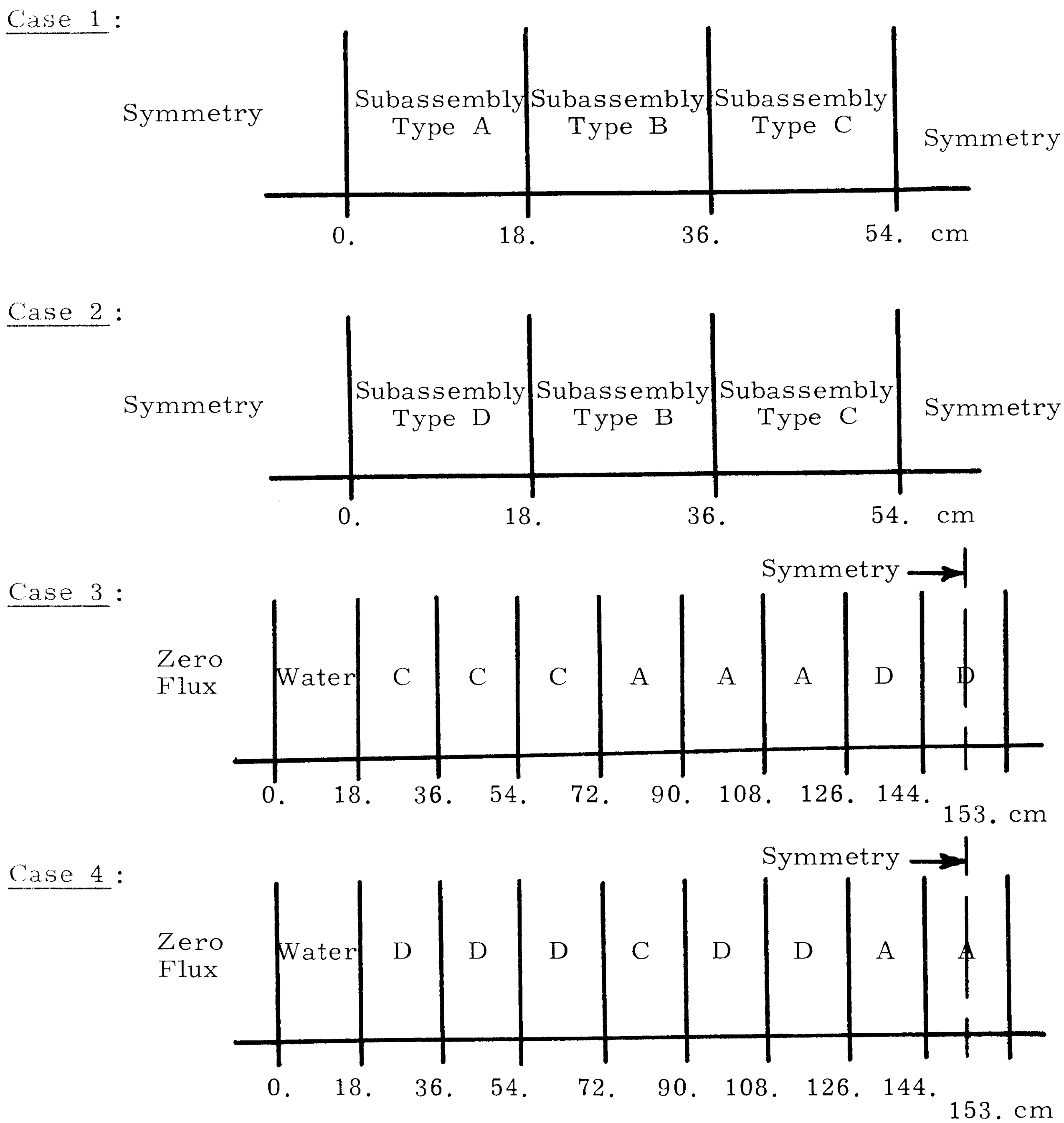


Figure 5.6. Geometry of the Four Case Studies Composed of Types of Subassemblies

The results of each case study are divided into the two approximation method categories as defined below.

1. The linear basis function approximation

   A. Linear FEM:

      (The linear finite element method using homogenized coarse mesh nuclear constants)

   B. Linear Synth:

      (The proposed approximation method using heterogeneous coarse mesh nuclear constants and detailed coarse mesh solutions)

2. The cubic Hermite basis function approximations

   A. Cubic FEM

   B. Cubic Synth

The results of the approximations in each category are compared to the reference solution by examining:

1. The converged eigenvalues λ (k_{eff}) and their percent normalized eigenvalue error,

   $$\% \lambda = (\lambda_{Ref} - \lambda_{Conv})/\lambda_{Ref} \times 100\%$$

2. Composite graphs of the converged detailed group flux solutions $U_g(z)$ normalized to equivalent power levels

3. The fractional normalized power levels P(k) calculated for each 18-cm subassembly k by

$$P(k) = \frac{\int_{\text{Sub. k}} \sum_{g=1}^{G} \nu\Sigma_{fg}(z)\, U_g(z)\, dz}{\int_{\text{All Subs.}} \sum_{g=1}^{G} \nu\Sigma_{fg}(z)\, U_g(z)\, dz} \tag{5.2}$$

and their percent normalized errors

$$\%P(k) = (P(k)_{\text{Ref}} - P(k)_{\text{Conv}})/P(k)_{\text{Ref}} \times 100\% \tag{5.3}$$

5.3.1. Case 1: Three different subassemblies of Types A, B, and C with symmetry boundary conditions.

The graphical results of the one-group approximation methods for this case are shown in Figures 5.7 and 5.8, while the results of the two-group approximation methods are presented in Figures 5.9 - 5.12. Only the coarse mesh boundaries are labeled in the figures, which indicates that entire 18-cm subassemblies were used as the coarse mesh regions. Two-group results using only half-subassemblies as the coarse mesh regions are shown in Figures 5.13 - 5.16. The reference solutions were calculated using 150-mesh regions, as defined by the symmetric partitioning

$$5(1\text{ cm}) + 4(.5\text{ cm}) + 4(.25\text{ cm}) + 4(.125\text{ cm}) + 8(.0625\text{ cm})$$

in each of the three subassemblies. The converged approximation eigenvalues and fractional normalized subassembly power levels for the one- and two-group calculations are summarized in Table 5.6. The fractional powers, P(k), for each subassembly are listed in the reference solution column, while the percent errors, %P(k) are listed in the approximation columns.

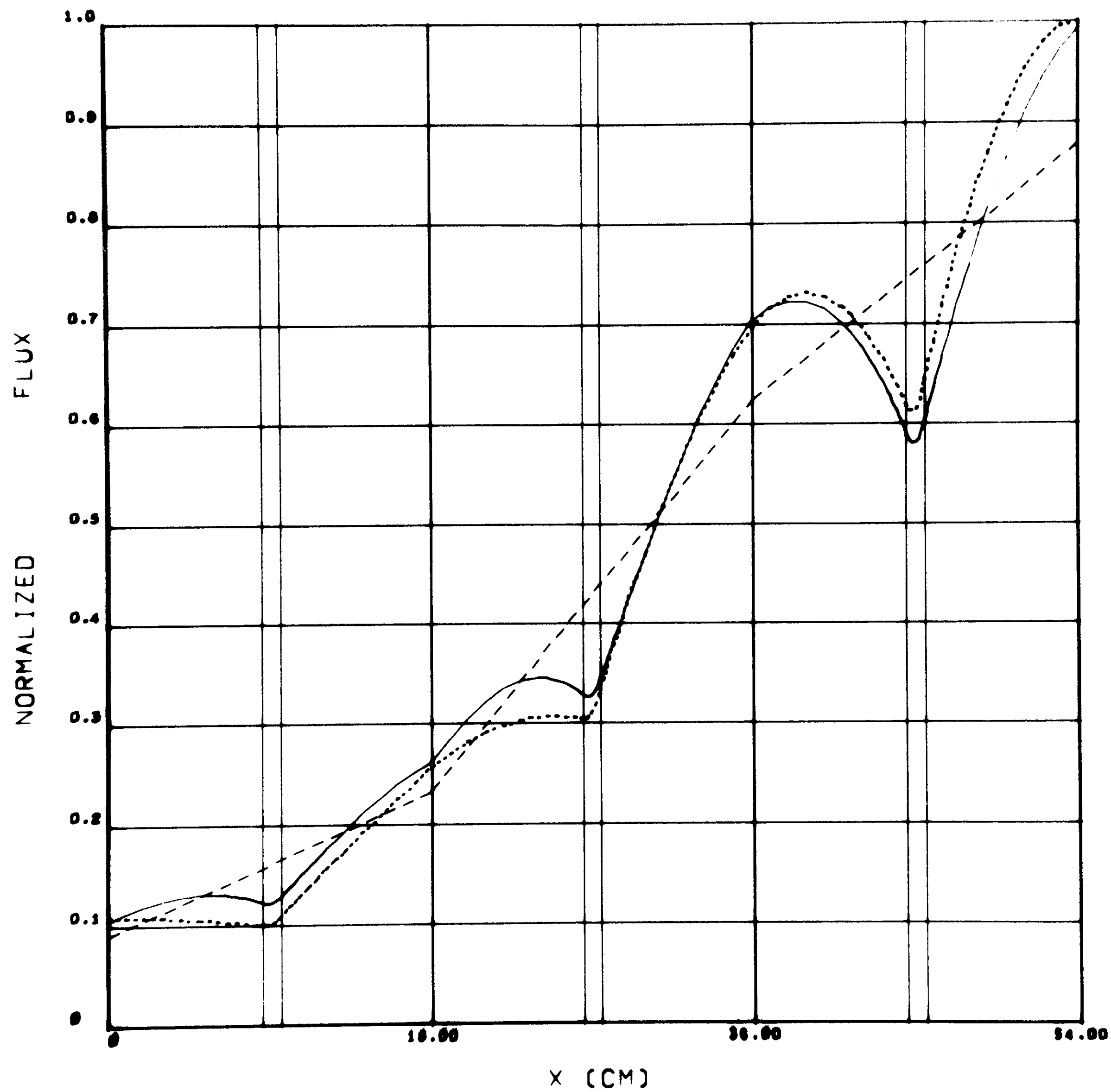


Figure 5.7. Case 1: One-Group Results Using Linear Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.559045
Linear FEM	— — — — —	.556943
Linear Synth	————	.557154

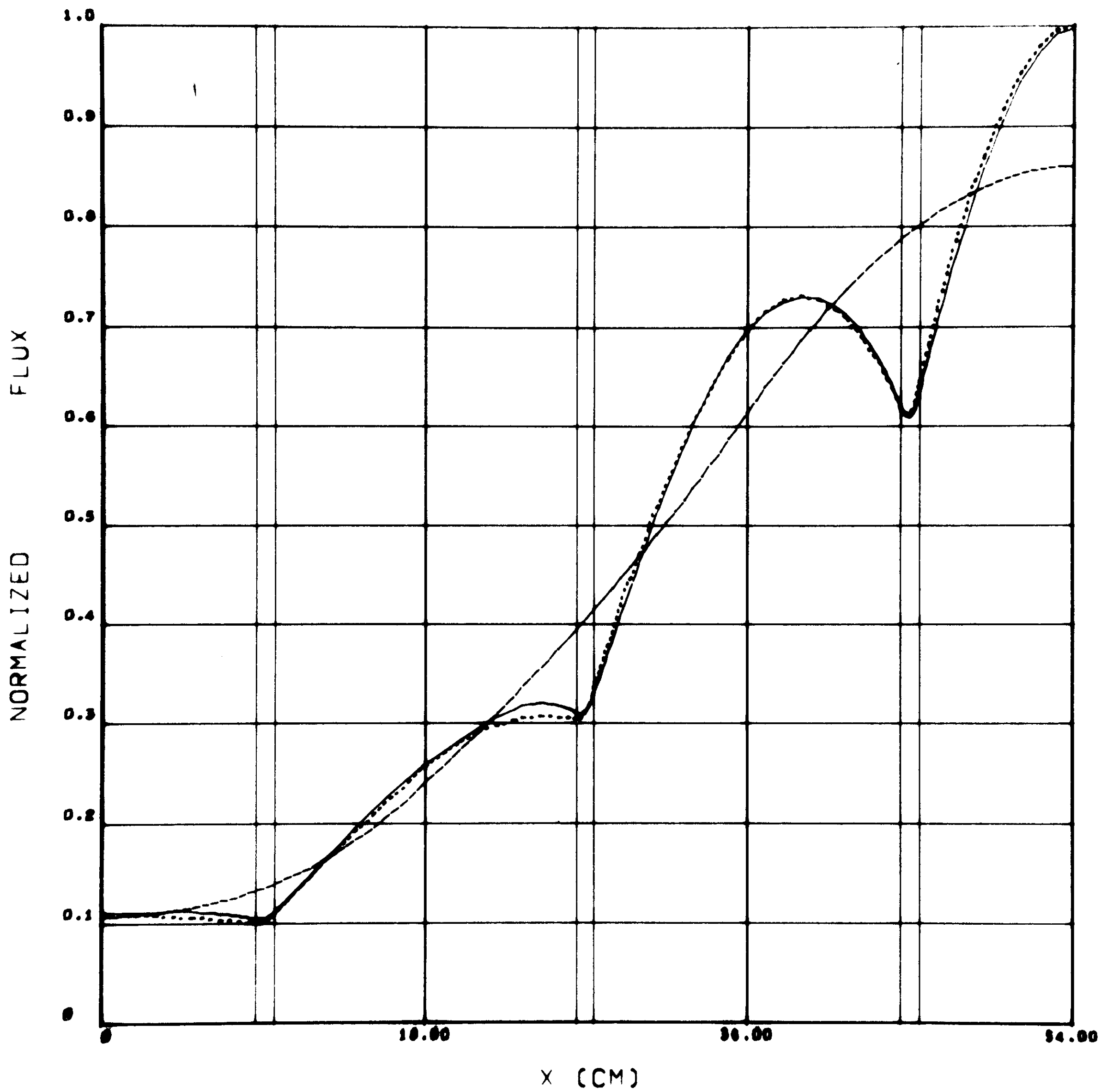


Figure 5.8. Case 1: One-Group Results Using Cubic Hermite Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.559045
Cubic FEM	— — — — —	.558647
Cubic Synth	————————	.558761

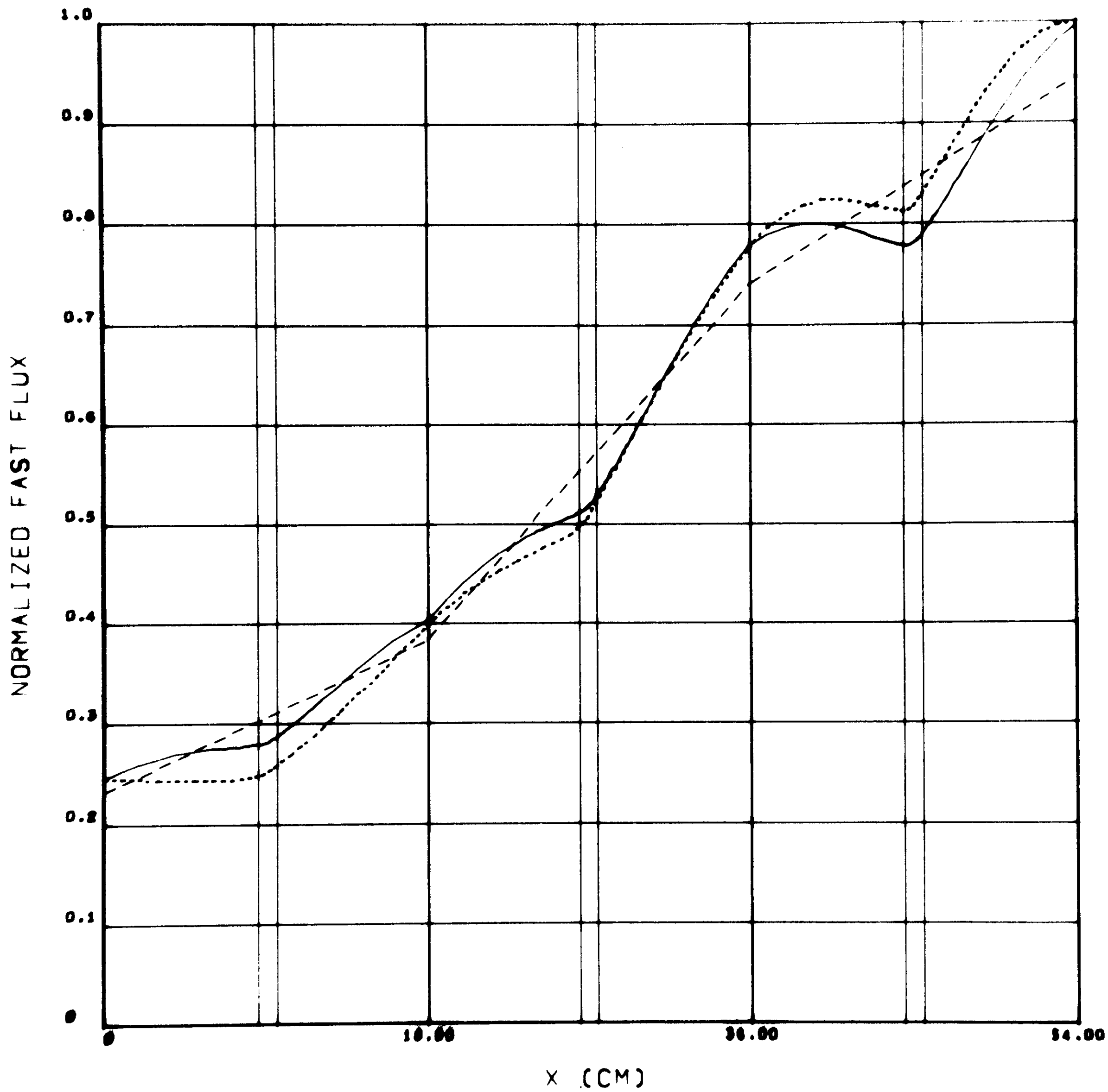


Figure 5.9. Case 1: Two-Group Fast Results Using Linear Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Linear FEM	– – – – –	.914489
Linear Synth	————	.915221

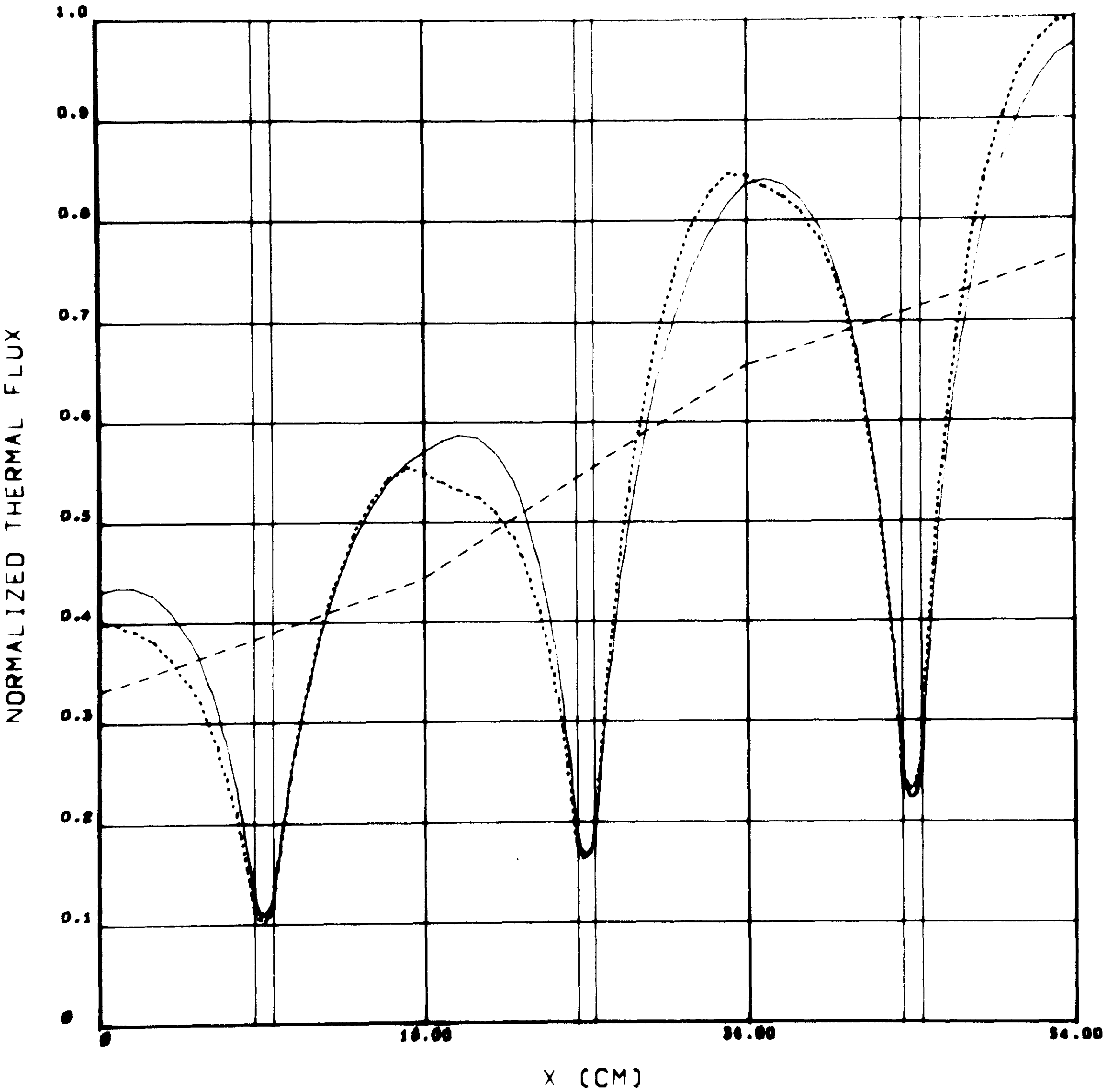


Figure 5.10. Case 1: Two-Group Thermal Results Using Linear Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Linear FEM	– – – – –	.914489
Linear Synth	————	.915221

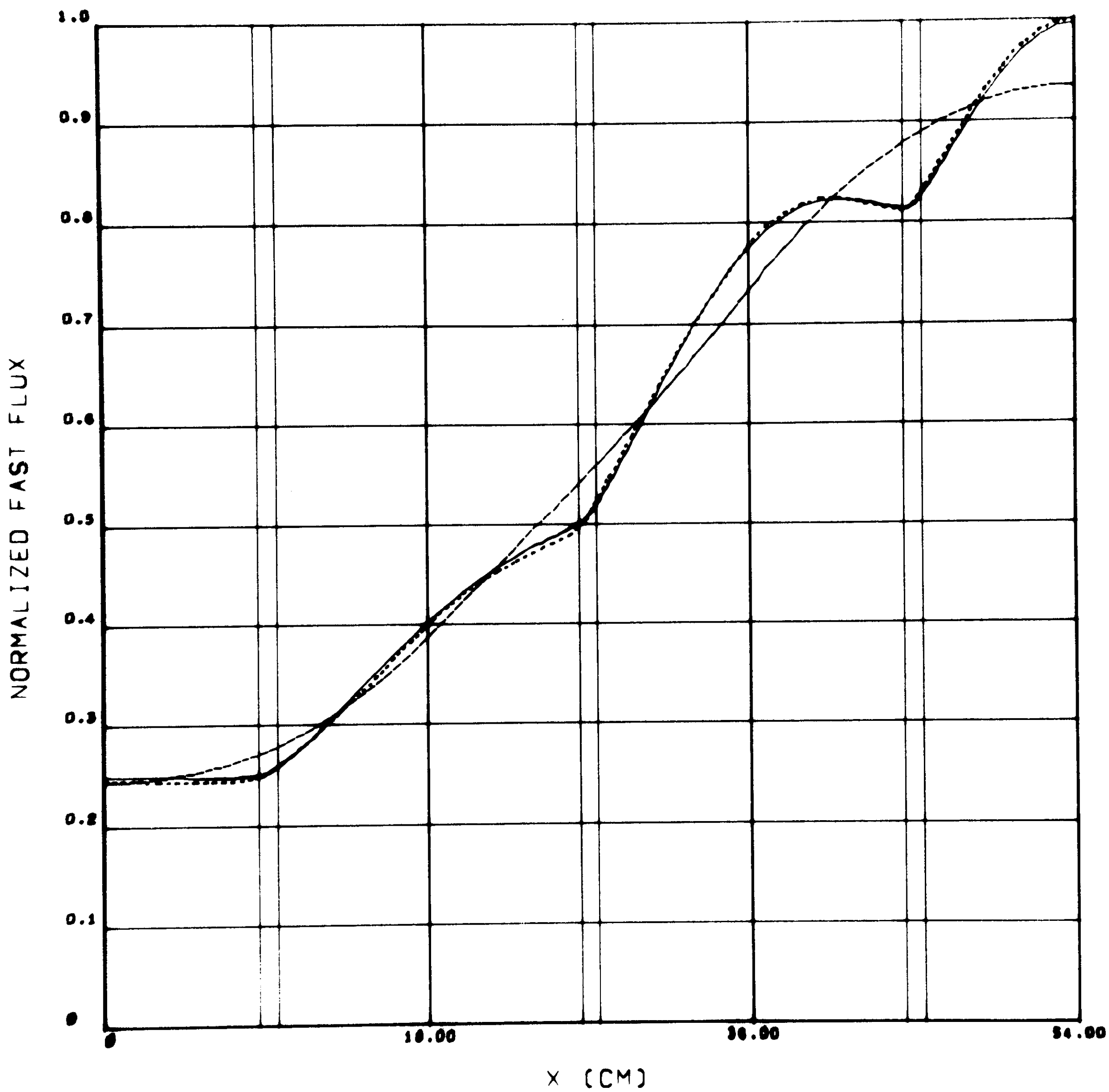


Figure 5.11. Case 1: Two-Group Fast Results Using Cubic Hermite Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Cubic FEM	— — — — —	.916717
Cubic Synth	————————	.917059

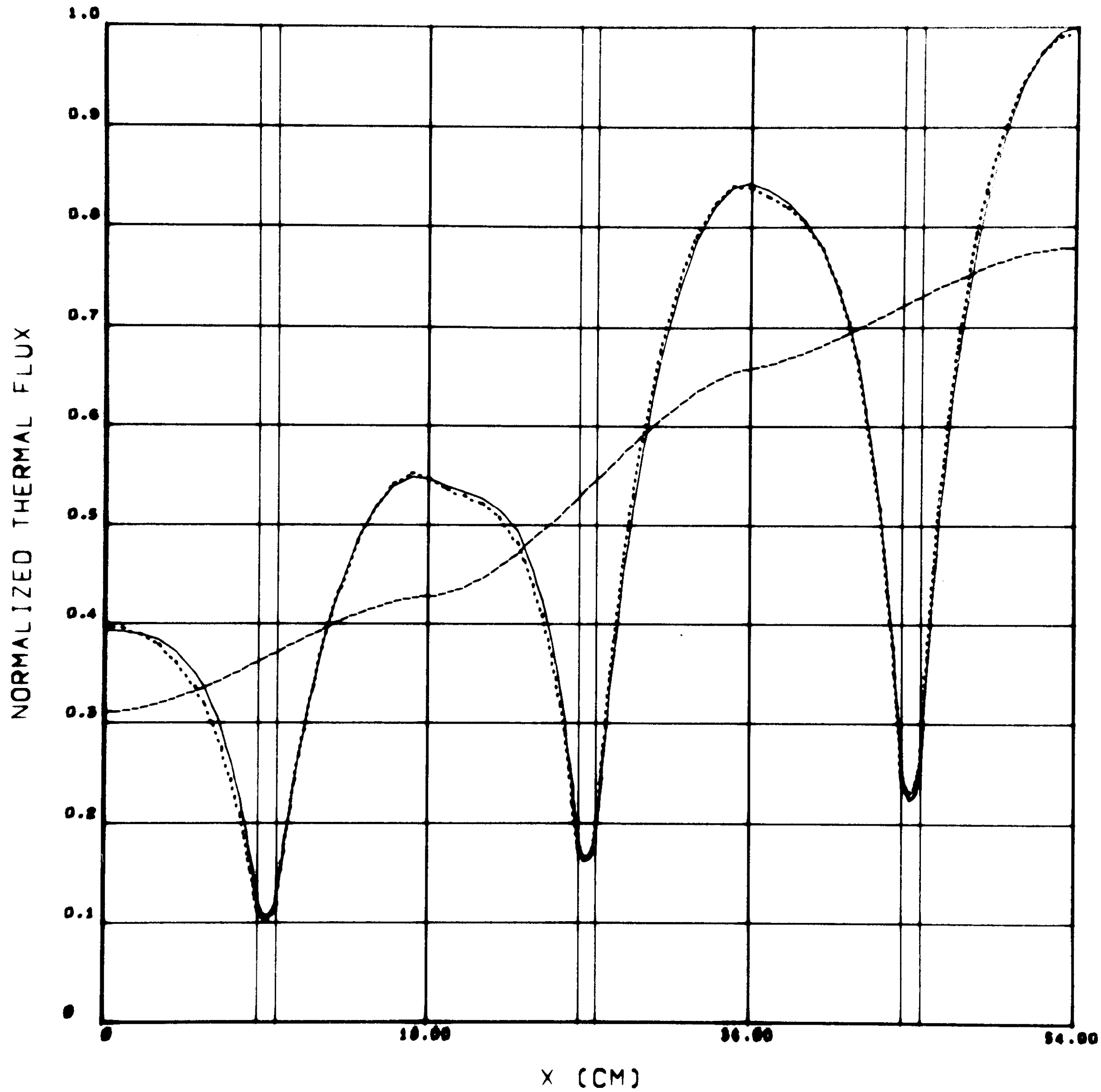


Figure 5.12. Case 1: Two-Group Thermal Results Using Cubic Hermite Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Cubic FEM	– – – – –	.916717
Cubic Synth	————	.917059

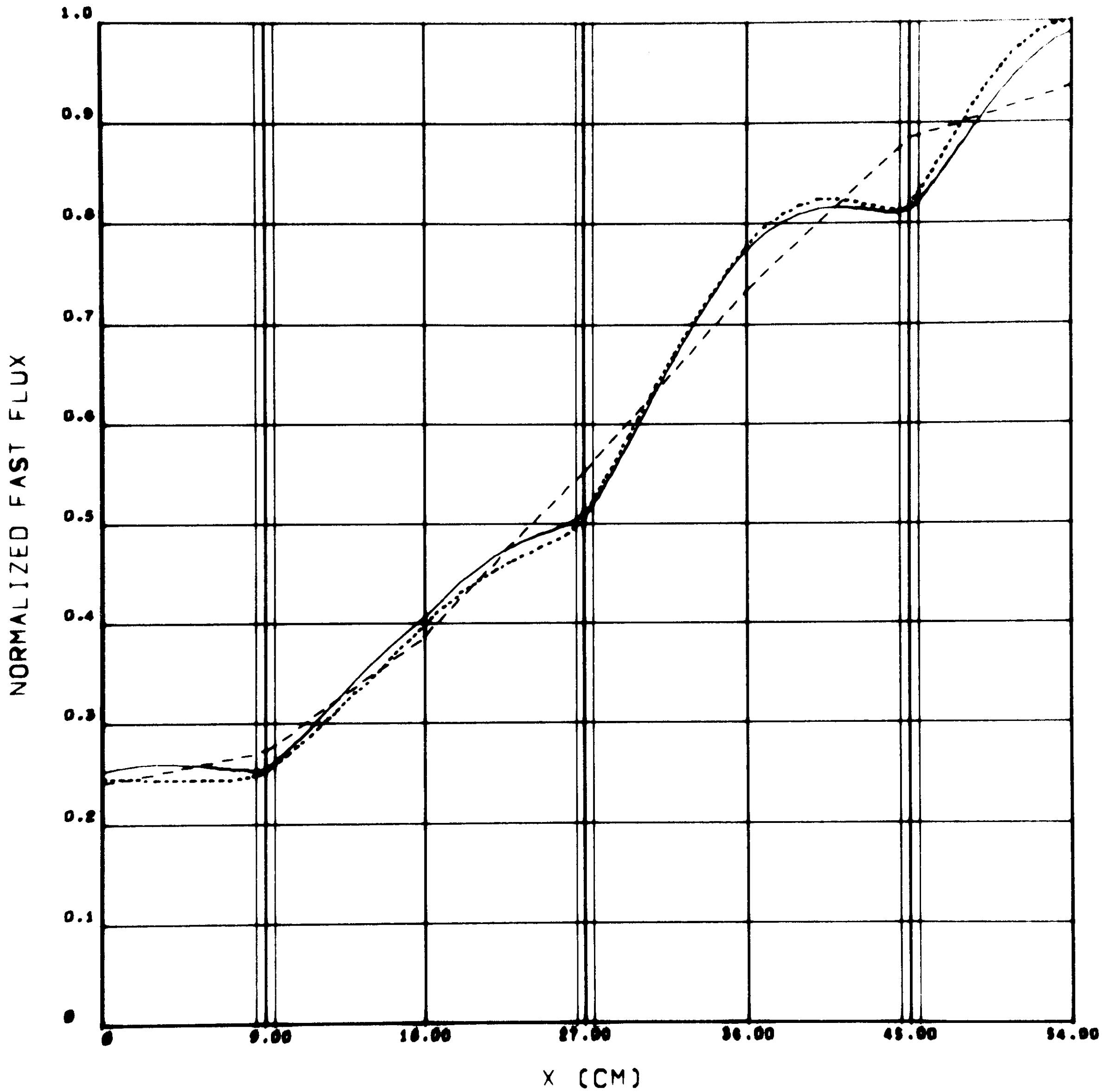


Figure 5.13. Case 1: Two-Group Fast Results Using Linear Basis Function Approximations and 9-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Linear FEM	— — — — —	.916356
Linear Synth	————————	.916427

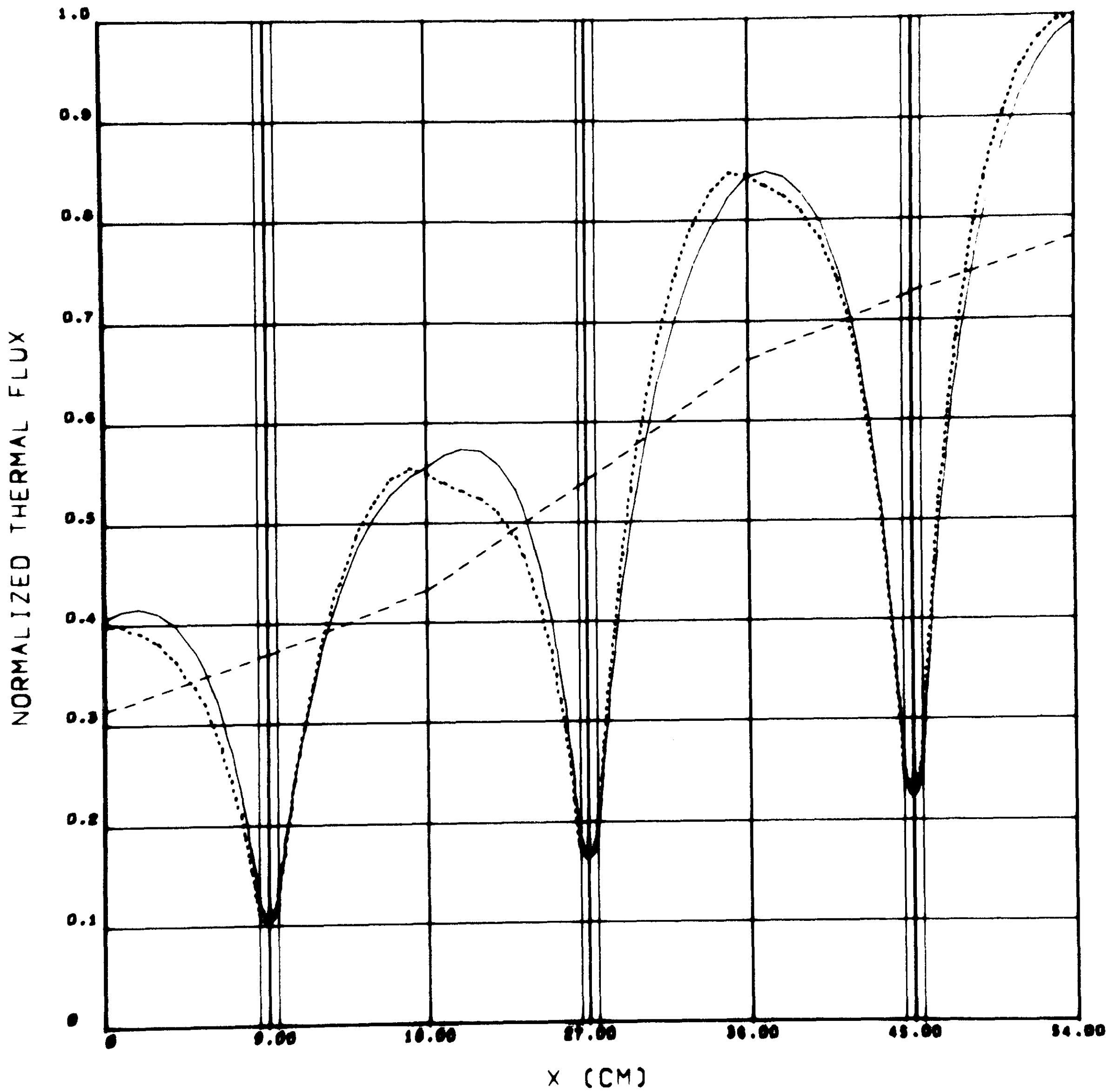


Figure 5.14. Case 1: Two-Group Thermal Results Using Linear Basis Function Approximations and 9-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Linear FEM	– – – – –	.916356
Linear Synth	————	.916427

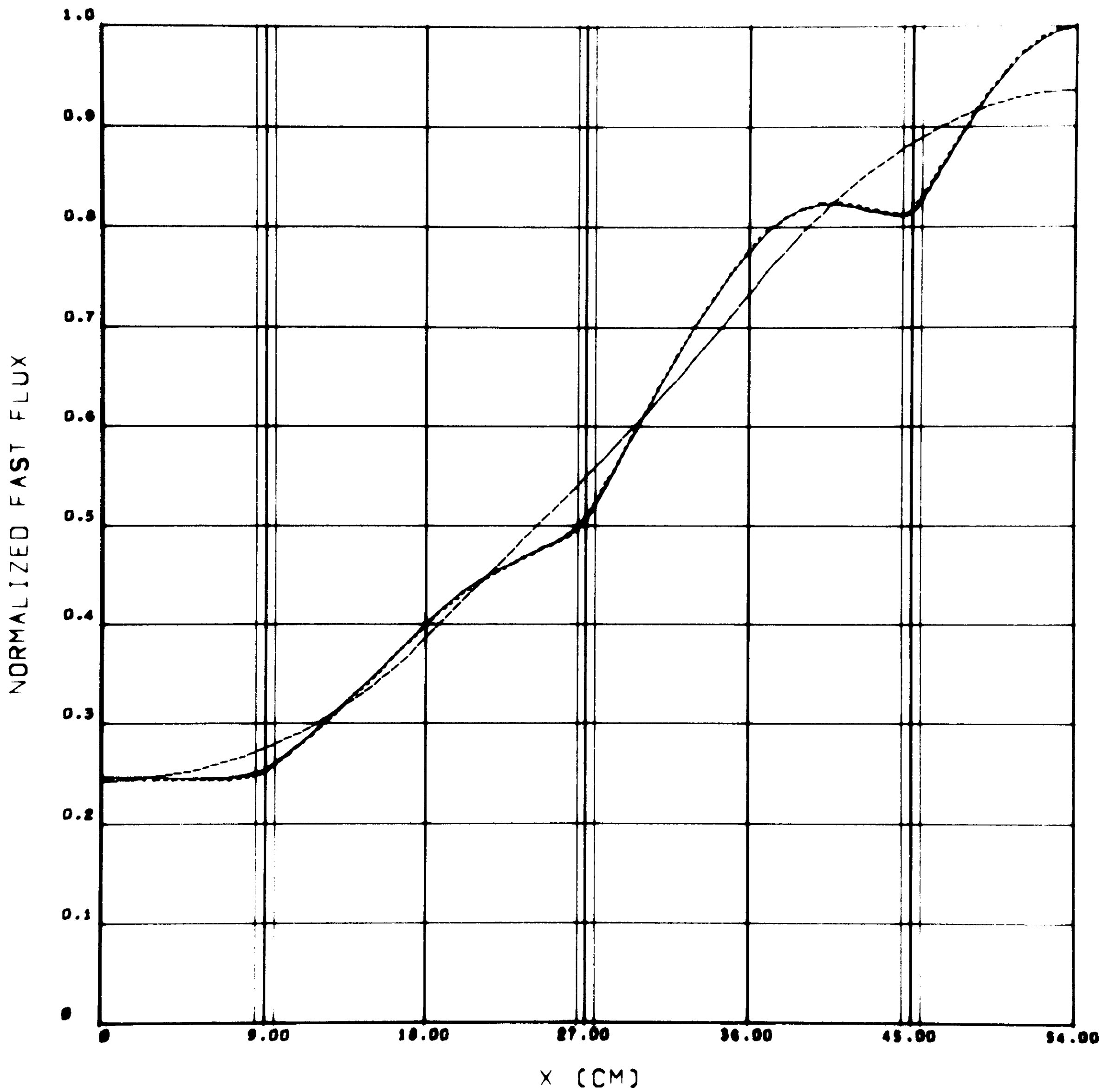


Figure 5.15. Case 1: Two-Group Fast Results Using Cubic Hermite Basis Function Approximations and 9-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Cubic FEM	– – – – –	.916669
Cubic Synth	————	.917294

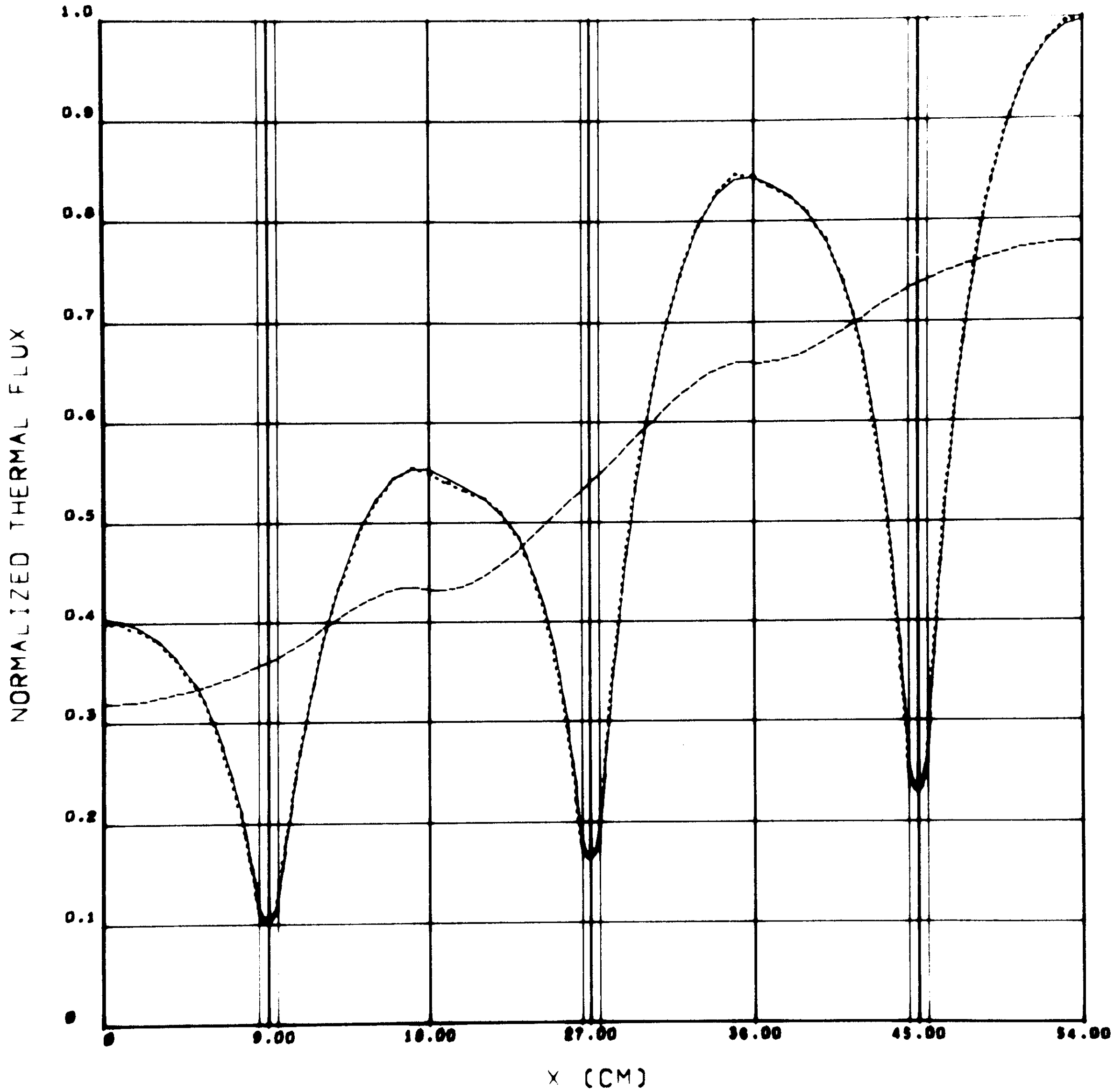


Figure 5.16. Case 1: Two-Group Thermal Results Using Cubic Hermite Basis Function Approximations and 9-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Cubic FEM	— — — — —	.916669
Cubic Synth	————————	.917294

Table 5.6. Results of Case 1.

Method / Results	Reference	Linear FEM	Linear Synth	Cubic FEM	Cubic Synth
ONE-GROUP RESULTS:					
λ	.559045	.556943	.557154	.558647	.558761
%λ	--	.376%	.338%	.072%	.051%
P(1)	.084	-12.1%	-11.9%	-3.07%	-2.44%
P(2)	.294	-4.29%	-3.93%	-.241%	-.434%
P(3)	.622	+3.67%	+3.47%	+.528%	+.534%
TWO-GROUP RESULTS:					
λ	.917267	.914489	.915221	.916717	.917059
%λ	--	.302%	.223%	.060%	.023%
P(1)	.134	-6.93%	-5.63%	-1.43%	-1.13%
P(2)	.315	-1.63%	-1.39%	-.072%	-.120%
P(3)	.549	+2.63%	+2.17%	+.391%	+.347%
Two-Group Results Using Half-Subassembly Mesh Regions					
λ	.917267	.916356	.916427	.916669	.917294
%λ	--	.093%	.092%	.065%	.003%
P(1)	.134	-2.38%	-2.69%	-1.51%	-.461%
P(2)	.315	-.475%	-.602%	-.063%	-.047%
P(3)	.549	+.851%	-2.63%	-3.22%	+1.40%

It is apparent from these results that the proposed approximation methods, and in particular the method utilizing the cubic Hermite basis functions, approximate to a high degree of accuracy the detailed reference spatial flux. Comparison of the eigenvalue and fractional power results in Table 5.6 indicates that comparable if not superior measurements are obtained using the proposed methods in this case.

It is interesting to note the effects of employing the given subassembly heterogeneous nuclear constants rather than subassembly homogenized nuclear constants for use in the finite element method calculations. Under such conditions, the finite element method becomes identical to the proposed methods in which the heterogeneous nuclear constants and constant or flat subassembly solutions are used. Two-group calculations using the cubic Hermite approximation method were performed for Case 1 and are presented in Figures 5.17 and 5.18. This scheme was found to give very poor detailed flux results, converge to an eigenvalue 21% in error, and yield an average of 20% error in the fractional normalized power levels in each subassembly. This example clearly illustrates the necessity for the use of homogenized constants in the finite element method, or equivalently, the importance of the subassembly detailed solutions in the proposed approximations.

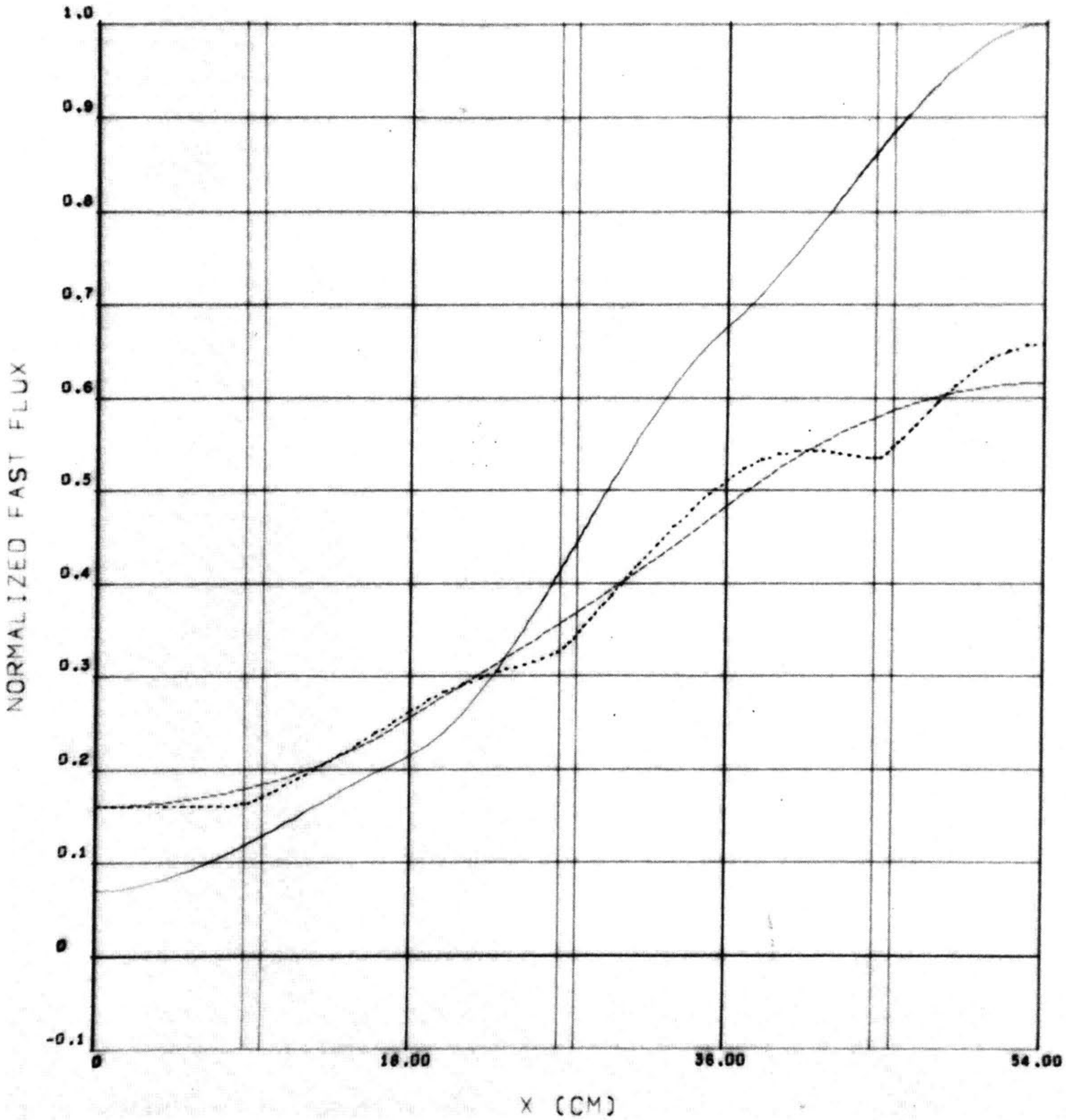


Figure 5.17. Case 1: Two-Group Fast Results Using Cubic Hermite Finite Element Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Cubic FEM + Homogenized Consts.	— — — — —	.916717
Cubic FEM + Detailed Consts.	————	.720422

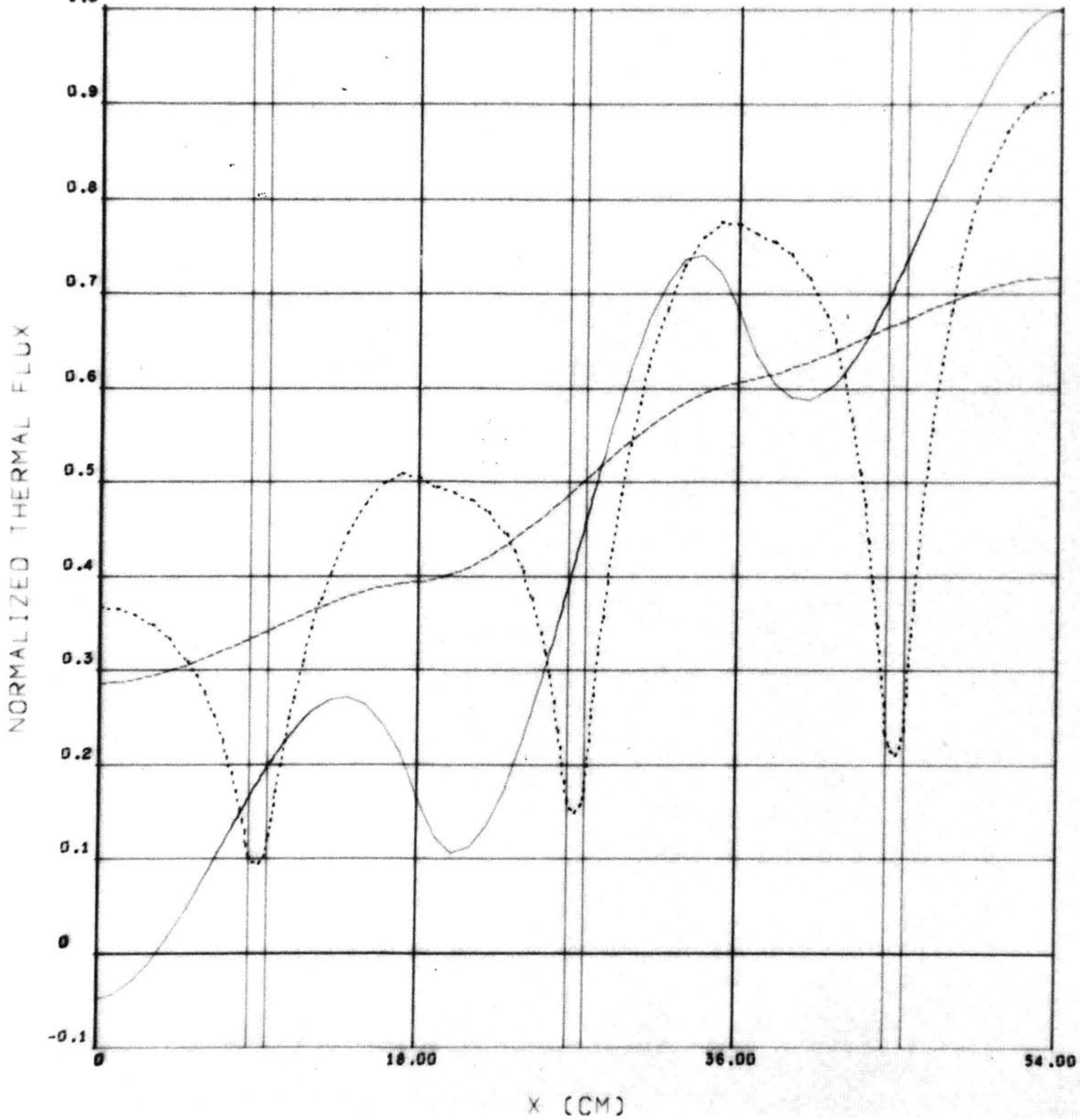


Figure 5.18. Case 1: Two-Group Thermal Results Using Cubic Hermite Finite Element Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.917267
Cubic FEM + Homogenized Consts.	– – – – –	.916717
Cubic FEM + Detailed Consts.	————	.720422

5.3.2. Case 2: Three different subassemblies of Types D, B, and C with symmetric boundary conditions.

The results of the two-group approximations for Case 2 are presented in Figures 5.19 - 5.22, where entire subassemblies were taken as the coarse mesh regions. The reference solutions were calculated using the same reference mesh geometry as in Case 1. The converged eigenvalues and fractional normalized power levels in each subassembly are summarized in Table 5.7. These results better illustrate the superiority of the cubic Hermite basis function approximations over the linear basis function approximations, and the superiority of the proposed approximations over the finite element method in all aspects.

Table 5.7. Two-Group Results of Case 2.

Method / Results	Reference	Linear FEM	Linear Synth	Cubic FEM	Cubic Synth
λ	.969986	.965260	.970236	.966816	.969578
% λ	--	.487%	-.026%	.326%	.042%
P(1)	.381	+11.26%	+3.46%	+6.07%	+.643%
P(2)	.296	-11.59%	-5.59%	-3.08%	-.157%
P(3)	.322	-2.66%	+1.03%	-4.34%	-.616%

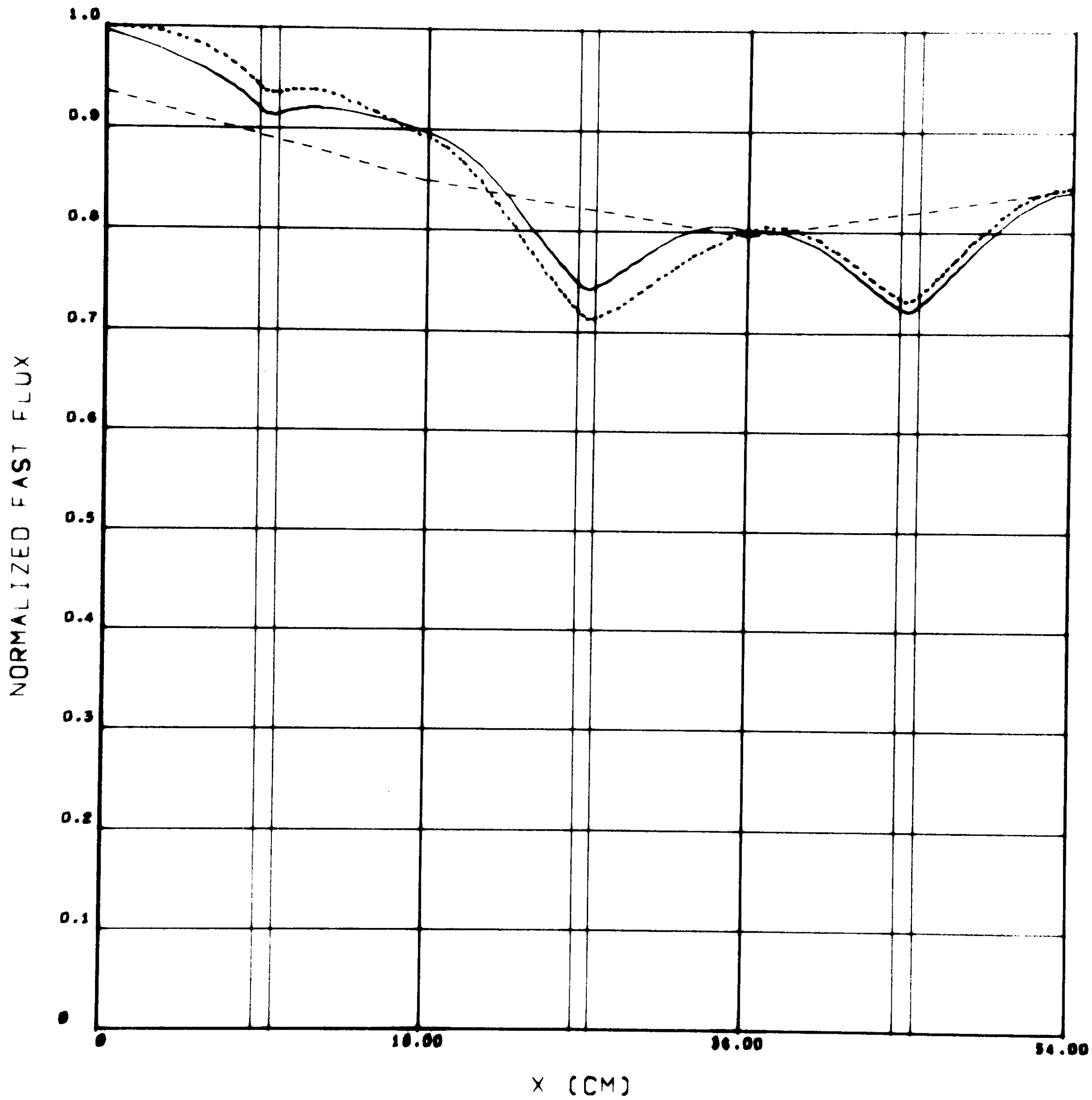


Figure 5.19. Case 2: Two-Group Fast Results Using Linear Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.969986
Linear FEM	– – – – –	.965260
Linear Synth	————	.970236

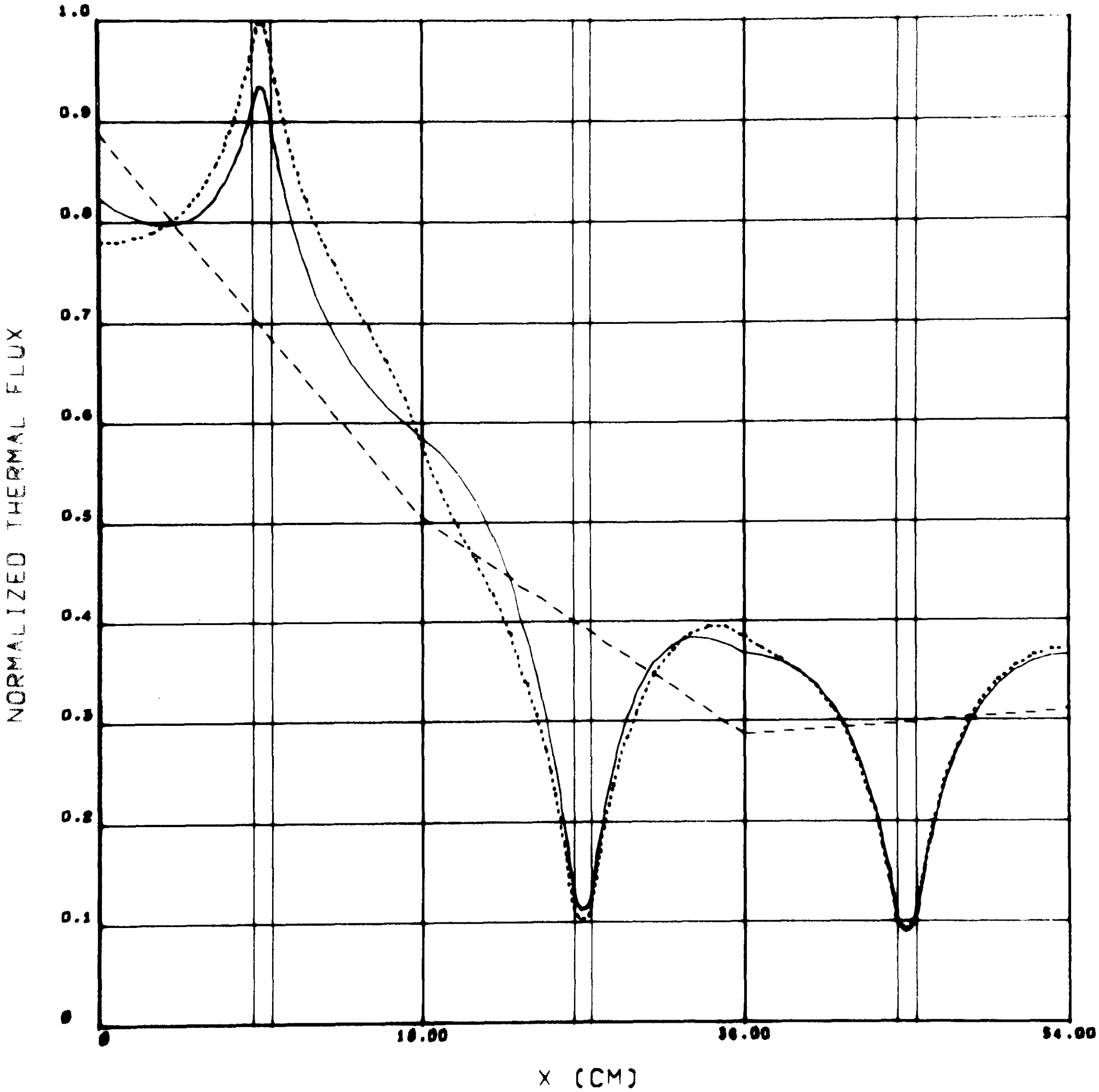


Figure 5.20. Case 2: Two-Group Thermal Results Using Linear Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.969986
Linear FEM	– – – – –	.965260
Linear Synth	————	.970236

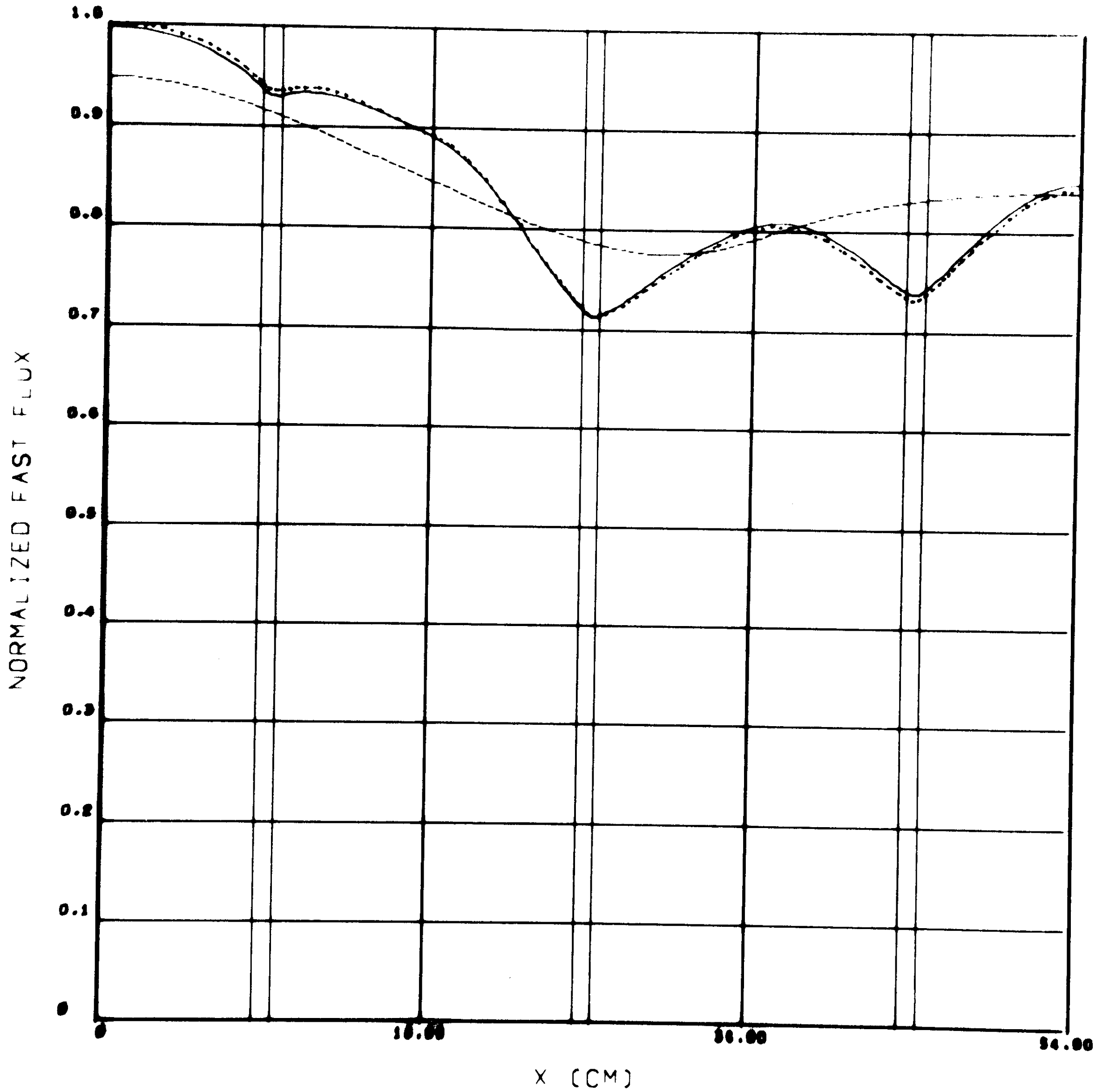


Figure 5.21. Case 2: Two-Group Fast Results Using Cubic Hermite Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	. . . .	.969986
Cubic FEM	– – – – –	.966816
Cubic Synth	————	.969578

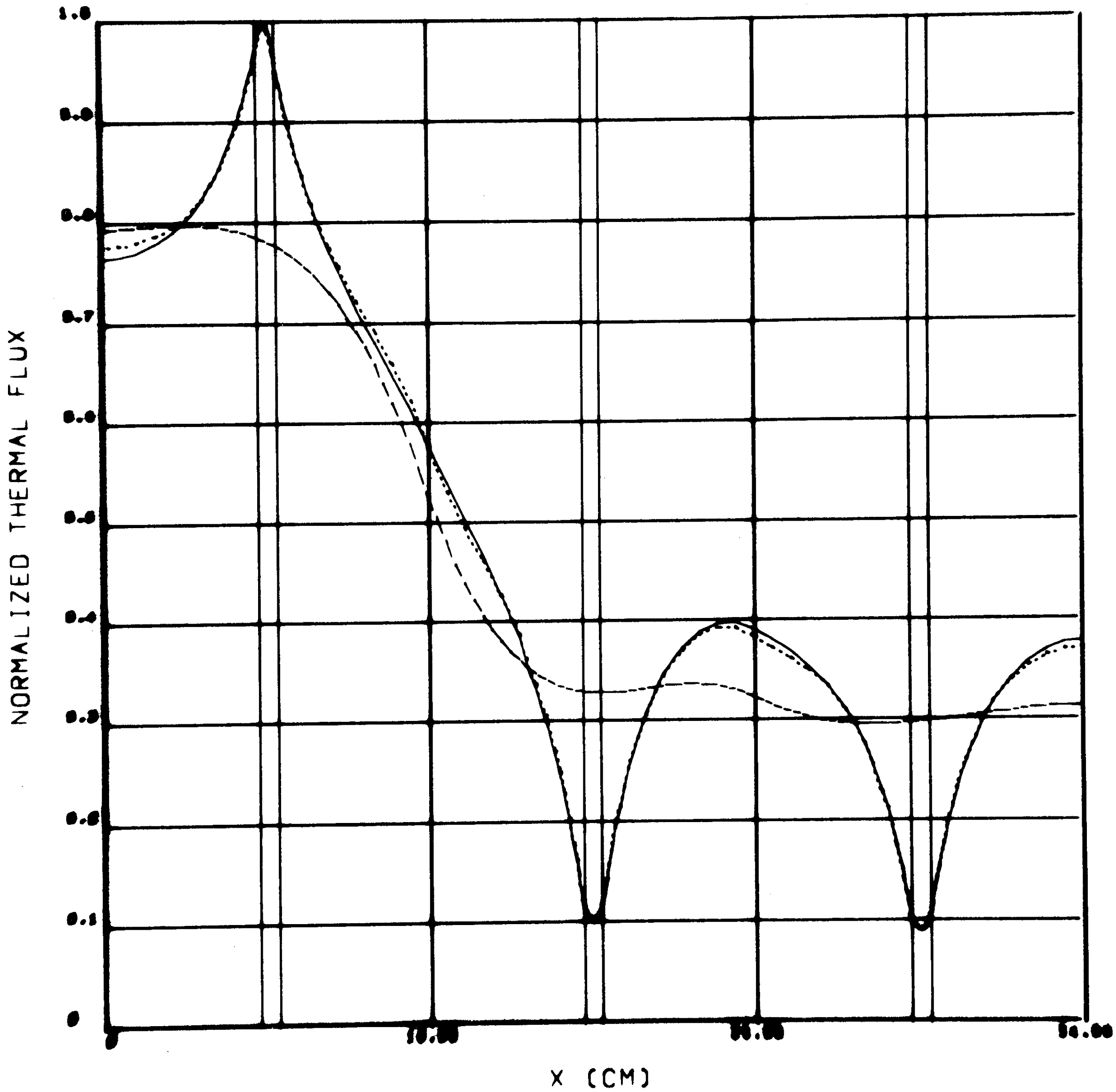


Figure 5.22. Case 2: Two-Group Thermal Results Using Cubic Hermite Basis Function Approximations and 18-cm Coarse Mesh Regions

Method		λ
Reference	....	.969986
Cubic FEM	— — — — —	.966816
Cubic Synth	————————	.969578

5.3.3. Case 3: Half-core reflected PWR composed of an 18-cm water reflector, the seven subassemblies C,C,C,A,A,A,D, and half of subassembly D. Zero flux boundary conditions are imposed outside the reflector, and symmetry is imposed in the center of the last D-type subassembly.

The Case 3 results of the two-group approximations using full 18-cm coarse mesh regions in all but the last 9-cm region are presented in Figures 5.23 - 5.26, and summarized in Table 5.8. The reference solutions were obtained using 198 mesh regions given by the symmetric partitioning

$$2(2\text{ cm}) + 2(1\text{ cm}) + 4(.5\text{ cm}) + 2(.25\text{ cm}) + 2(.25\text{ cm})$$

in each of the subassemblies, and 18 (1 cm) regions in the reflector.

The use of many subassemblies containing absorption rods throughout the reactor, except in the center subassemblies where water channels are present, results in central peaked fluxes with large gradients and, by comparison, a relatively small thermal neutron peak in the reflector.

Both coarse mesh methods were found to overestimate the flux in the subassemblies near the reflector, and underestimate the flux in the central subassembly regions regardless of the type of basis function approximations used. The larger inaccuracies of the linear basis function methods can be in part attributed to the fact that these methods cannot approximate the peaked thermal flux in the reflector, and result in large flux values in the subassemblies nearest the reflector. The cubic Hermite basis function approximations, however, are better able to approximate both the thermal flux reflector peak and the complex

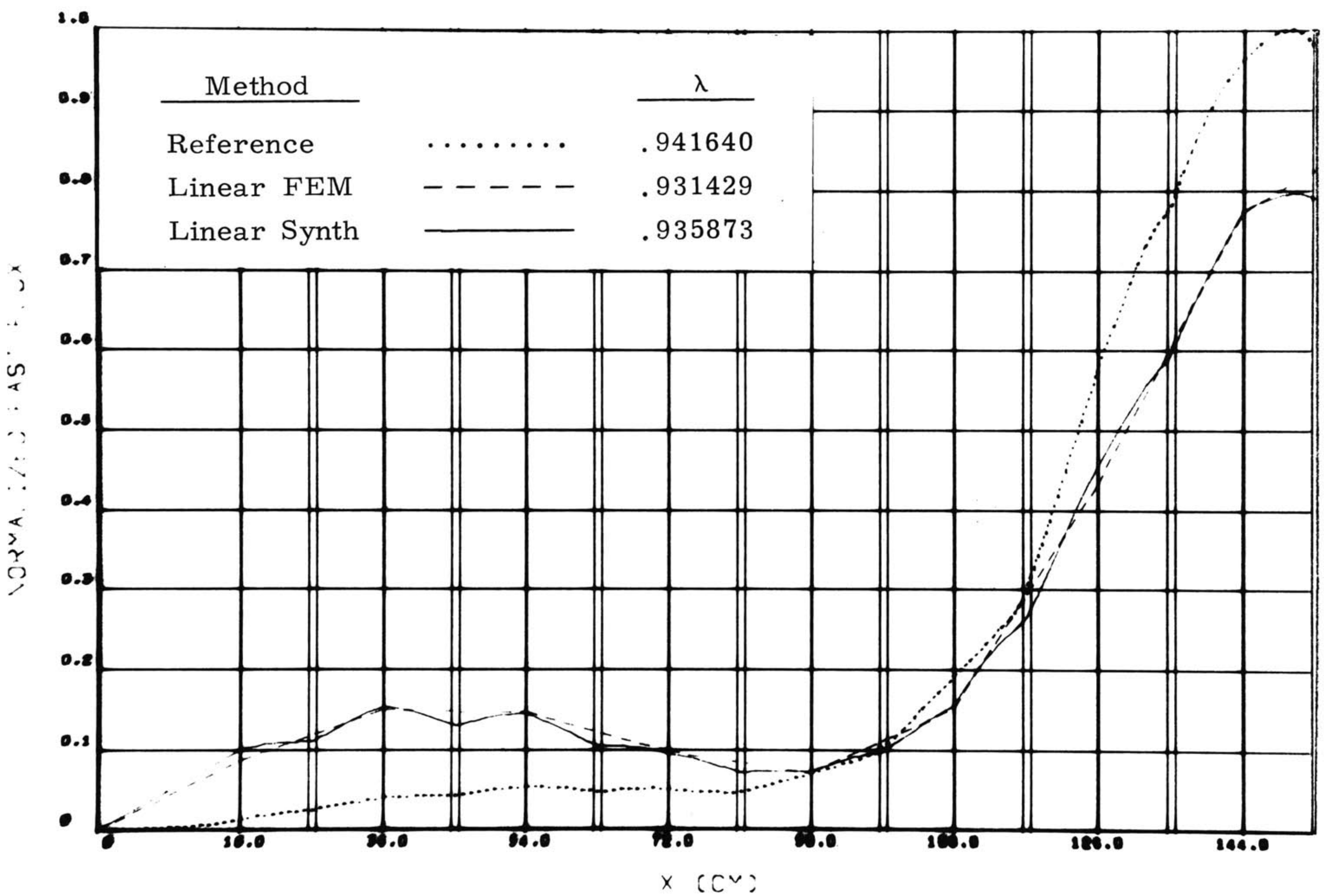


Figure 5.23. Case 3: Two-Group Fast Results Using Linear Basis Function Approximations and 18-cm Coarse Mesh Regions

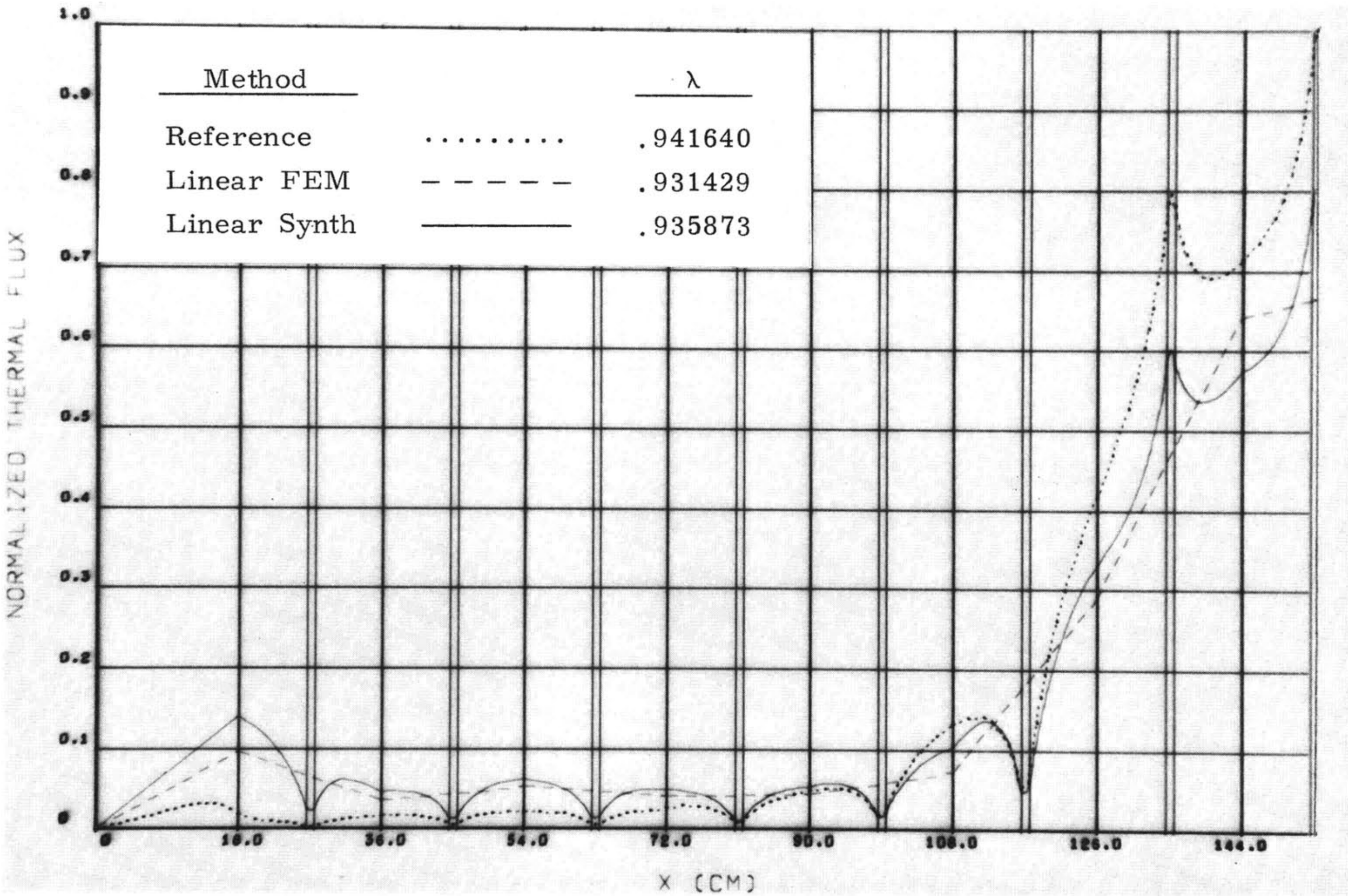


Figure 5.24. Case 3: Two-Group Thermal Results Using Linear Basis Function Approximations and 18-cm Coarse Mesh Regions

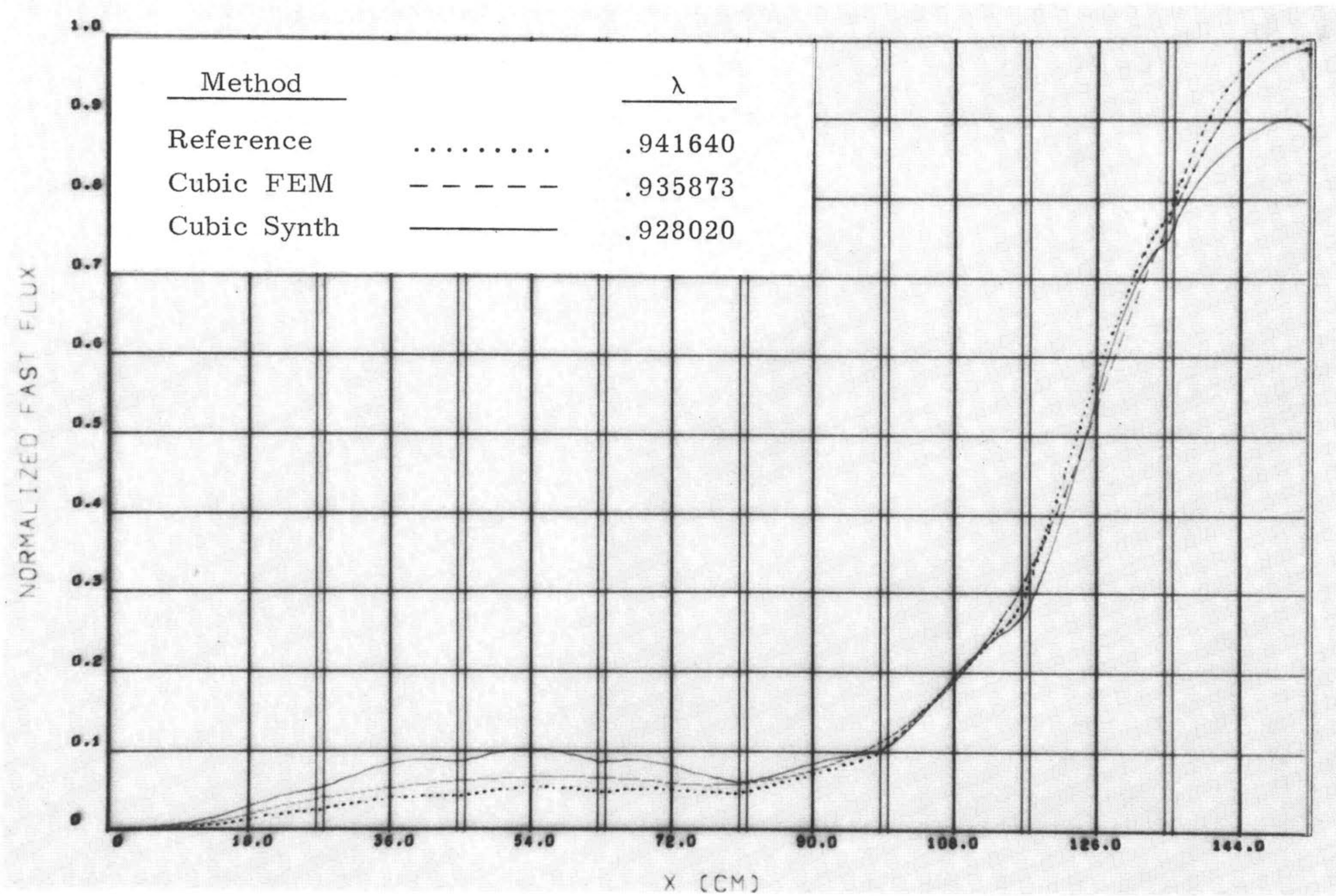


Figure 5.25. Case 3: Two-Group Fast Results Using Cubic Hermite Basis Function Approximations and 18-cm Coarse Mesh Regions

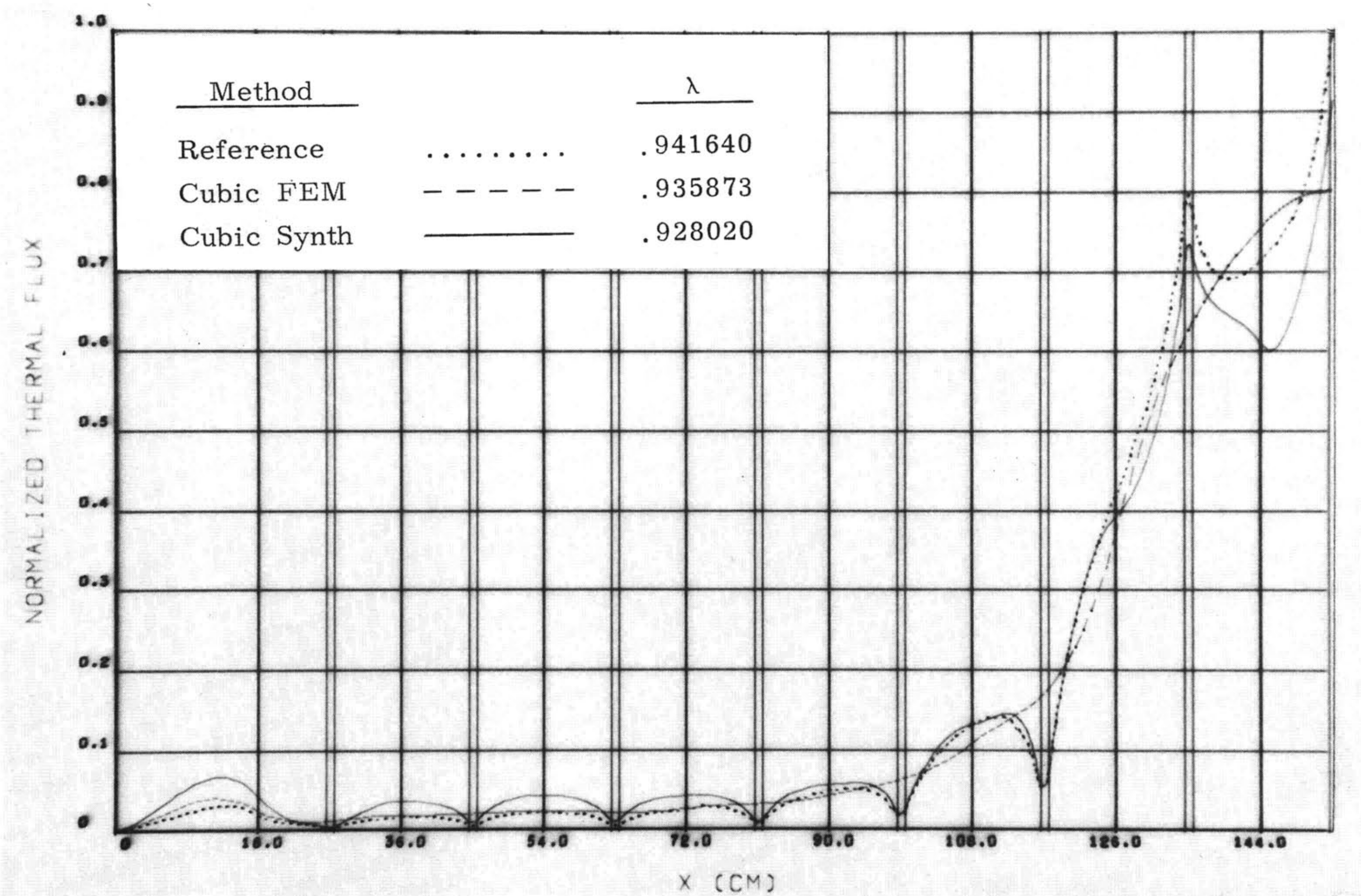


Figure 5.26. Case 3: Two-Group Thermal Results Using Cubic Hermite Basis Function Approximations and 18-cm Coarse Mesh Regions

Table 5.8. Results of Case 3.

Method / Results	Reference	Linear FEM	Linear Synth	Cubic FEM	Cubic Synth
λ	.941640	.931429	.938561	.935873	.928020
% λ	--	1.08%	.32%	.61%	1.44%
P(1)	.01699	-461.%	-514.%	-37.4%	-85.1%
P(2)	.02513	-192.%	-179.%	-32.8%	-98.6%
P(3)	.02890	-149.%	-133.%	-26.0%	-79.6%
P(4)	.0224	-50.2%	-36.9%	-13.5%	-31.5%
P(5)	.04969	8.52%	8.24%	-2.78%	-7.63%
P(6)	.1465	7.47%	13.7%	-.494%	.259%
P(7)	.4362	26.4%	22.9%	5.01%	8.40%
P(8)	.2740	18.6%	20.1%	1.99%	13.1%

neutron leakage across the core, and give better results. Table 5.8 indicates that the cubic Hermite basis function approximations better approximate the detailed reference solutions, and that results obtained using the cubic Hermite finite element method were for this case better than those obtained using either of the proposed approximations. The ability of these methods to approximate large thermal flux peaks in the reflector regions is considered in the next case.

5.3.4. Case 4: Half-core reflected PWR composed of an 18-cm water reflector, the seven subassemblies D,D,D,C,D,D,A, and half of subassembly Type A. Zero flux boundary conditions are imposed in the center of the last Type A subassembly.

The Case 4 geometry produces a large but detailed thermal flux in the half-core region and a large thermal peak in the reflector region, as seen from the results in Figures 5.27 - 5.30. The reference solutions were calculated using the reference mesh geometry as given in Case 3. The results of the approximations are summarized in Table 5.9.

The results show that the linear basis function approximations cannot approximate accurately the thermal flux reflector peak and result in large flux and fractional power errors in the subassemblies near the reflector. The cubic Hermite basis function approximations, on the other hand, are better able to approximate this thermal peak and result in much more accurate power levels, especially in the first subassembly region.

The Case 4 results typify the approximation accuracy of both the finite element method and the proposed approximation method. In general, the cubic Hermite basis function approximations are superior to the linear basis function approximations, and the proposed methods give comparable or superior results as compared to those obtained from the finite element method using the same class of basis functions. In this case, the proposed method using cubic Hermite basis functions was able to estimate the reference eigenvalue within 0.04%, closely approximate the detailed reference flux solution to within a few percent at all spatial points, and result in fractional normalized power levels in each subassembly with less than 5% error.

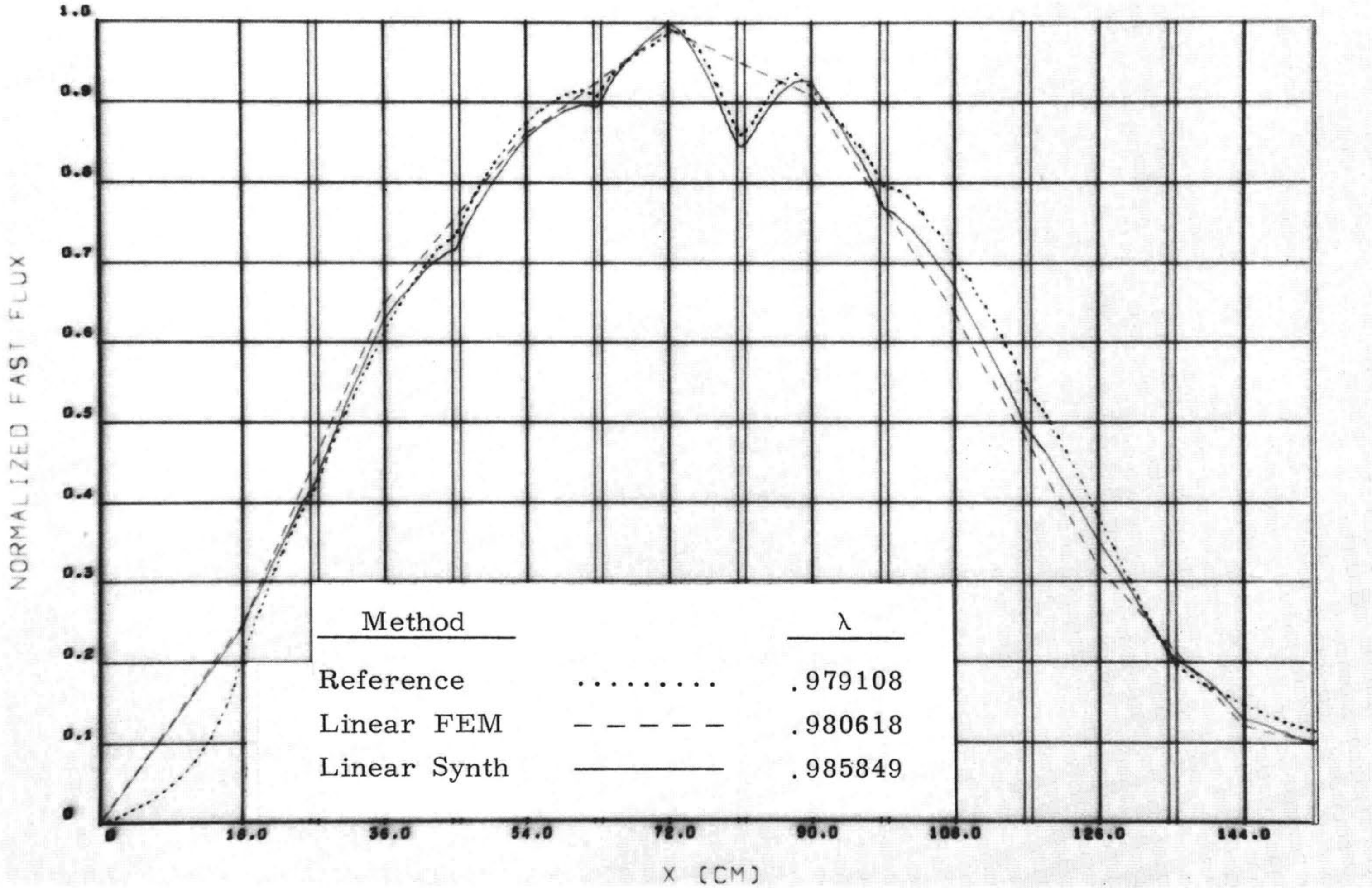


Figure 5.27. Case 4: Two-Group Fast Results Using Linear Basis Function Approximations and 18-cm Coarse Mesh Regions

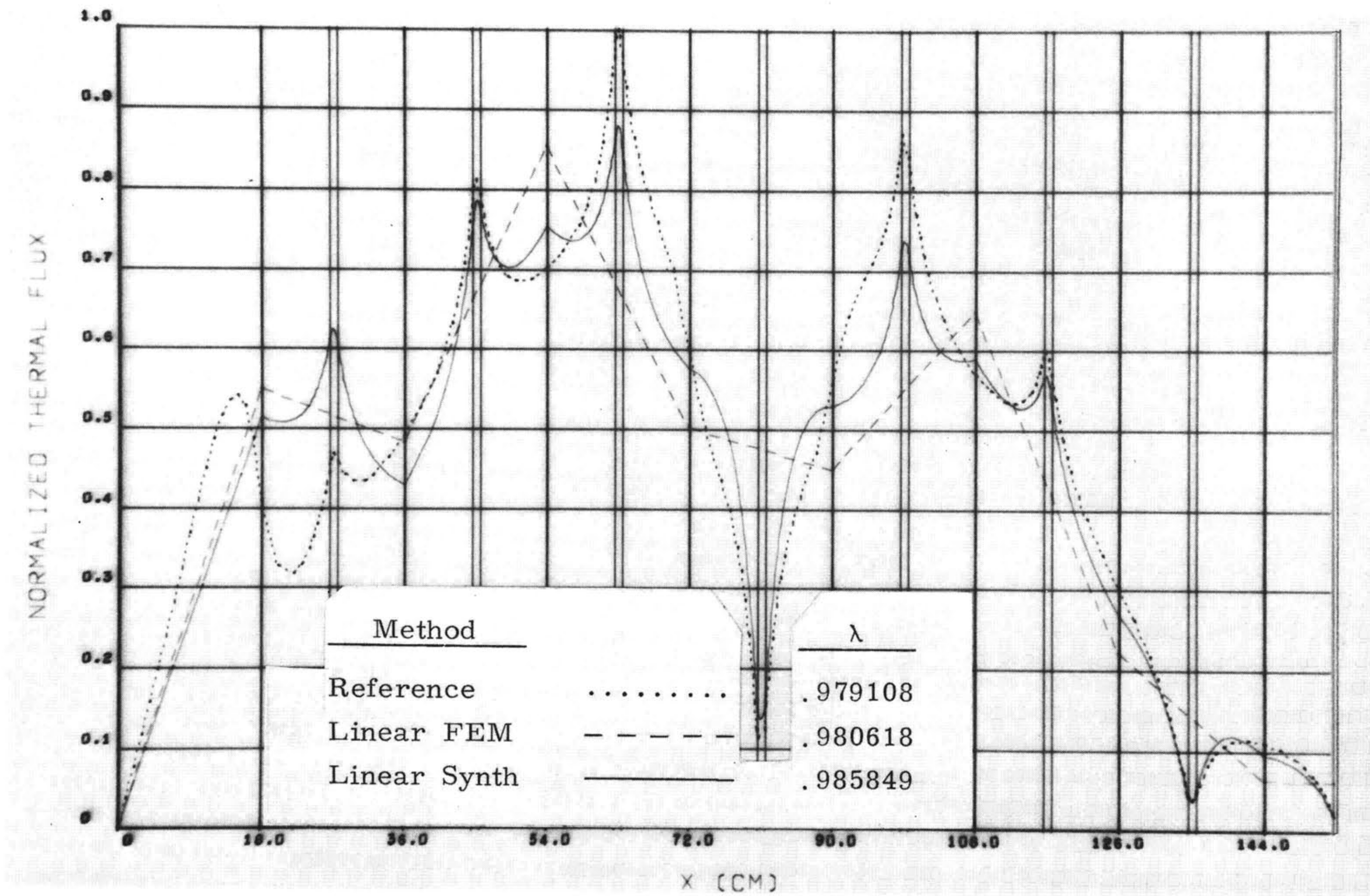


Figure 5.28. Case 4: Two-Group Thermal Results Using Linear Basis Function Approximations and 18-cm Coarse Mesh Regions

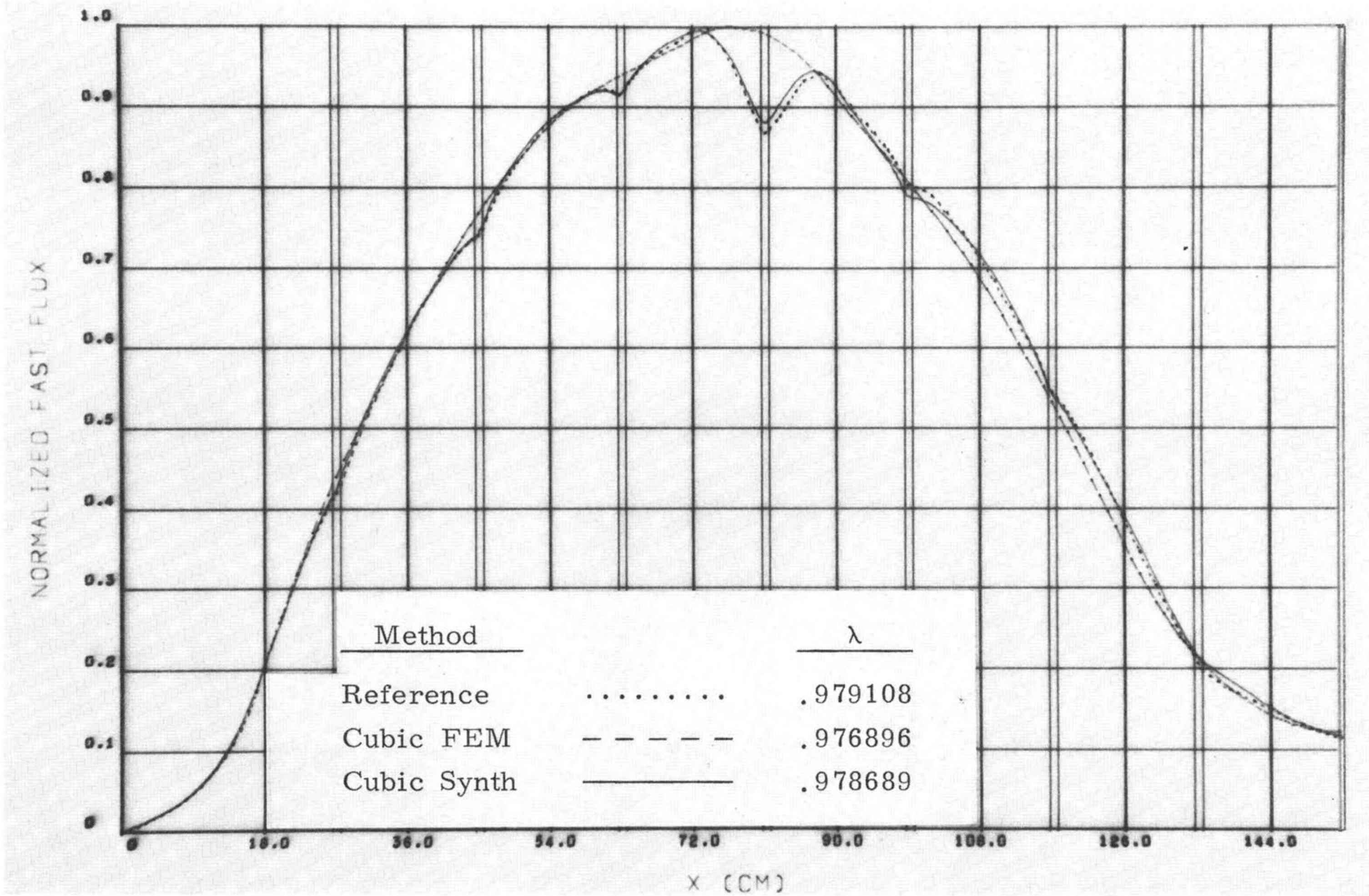


Figure 5.29. Case 4: Two-Group Fast Results Using Cubic Hermite Basis Function Approximations and 18-cm Coarse Mesh Regions

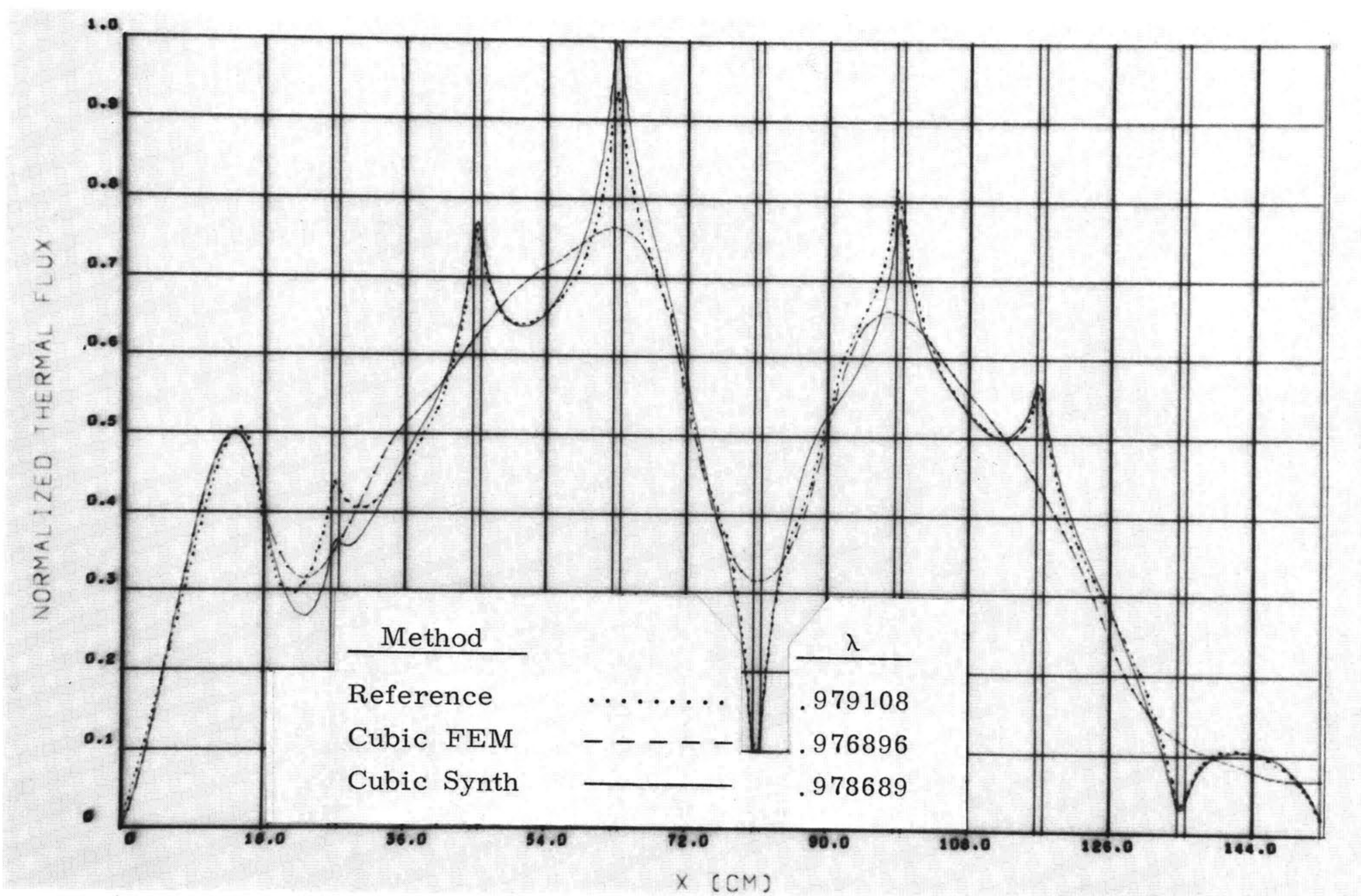


Figure 5.30. Case 4: Two-Group Thermal Results Using Cubic Hermite Basis Function Approximations and 18-cm Coarse Mesh Regions

Table 5.9. Results of Case 4.

Method / Results	Reference	Linear FEM	Linear Synth	Cubic FEM	Cubic Synth
λ	.979108	.980618	.985849	.976896	.978689
% λ	--	-.15%	-.69%	.22%	.04%
P(1)	.098	-23.4%	-21.4%	-13.3%	5.54%
P(2)	.160	-1.53%	2.39%	-.893%	-1.08%
P(3)	.193	11.5%	7.11%	1.83%	-3.35%
P(4)	.211	-19.5%	-10.6%	-7.23%	-1.60%
P(5)	.168	17.4%	10.3%	3.86%	3.79%
P(6)	.118	10.6%	4.96%	5.97%	-1.01%
P(7)	.039	1.45%	3.44%	1.65%	3.25%
P(8)	.010	18.1%	11.8%	2.30%	-2.72%

Chapter 6

CONCLUSIONS AND RECOMMENDATIONS

6.1. Characteristics of the Proposed Approximation Methods

The use of detailed subassembly flux solutions or other a priori flux shapes directly in the spatial shape or trial function form of flux approximations in reactor physics has resulted in many coarse mesh approximation schemes which are classified in the broad area of overlapping multichannel synthesis. The proposed approximation methods are similar to existing synthesis methods of this kind, but are unique in that they reduce to conventional and well understood approximation methods in regions where little or no spatial flux information is given, or in completely homogeneous regions. In contrast, the overlapping synthesis methods proposed to date do not. This characteristic is especially important in calculations involving homogeneous regions, of which reflector regions are a prime example.

The proposed approximations are very similar to coarse mesh finite element method approximations in which detailed flux behavior has been used to flux-weight the nuclear constants in each region. The methods are conceptually different and become equivalent only when all of the coarse mesh regions are homogeneous.

The matrix equations resulting from the use of the proposed methods are identical in form to those resulting from the finite element method utilizing similar basis functions. In addition, the matrix elements of the proposed methods are curiously different from those of the finite element methods using detailed flux-weighted nuclear constants. Although the spatial mass and stiffness matrices of the proposed methods for each group have been proven to be positive definite only for the case of Galerkin flux weighting, the use of adjoint weighting in all of the cases considered did not alter these properties. In addition, the proposed methods were found always to converge to a positive eigenvalue and to flux shapes which were everywhere positive.

The numerical results indicate that the proposed methods are able to predict accurate criticality or k_{eff} measurements and regional power levels as well as to approximate the reference detailed flux shapes for each group with a high degree of accuracy. The results indicate that in general, use of the proposed methods results in superior criticality estimates over those obtained by the use of the finite element method with flux-weighted constants; this behavior was observed for each type of basis function approximation. Moreover, each of the proposed methods is in general vastly superior to its finite element method counterparts in approximating the actual detailed flux behavior and regional as well as total power levels.

Detailed flux behavior could be reintroduced into the results of the homogenized finite element methods by normalizing the detailed subassembly solutions in each coarse mesh region to match the power levels of the converged results in each region. The detailed solutions resulting from such a procedure would be discontinuous at the region boundaries and may, to some extent, exhibit the fine flux structure present in the results of the proposed methods. However, the results are not expected to be as good as an approximation as those of the proposed methods, since the current coupling or diffusion approximation is not made until after the coarse mesh homogenization procedure.

6.2 Applicability and Limitations

Because the matrix forms of the equations which result from the use of the proposed methods are identical to those which result from the use of the finite element methods, the proposed approximations can be incorporated into existing finite element approximation schemes. Although additional integrations must be performed in the proposed methods, they can be reduced to sums of known products so that little additional computation time is required.

As in any coarse mesh approximation method, inaccurate results can occur when the coarse mesh region sizes chosen are too large. For a given region size, the accuracy of the results for any approximation scheme is unknown. The accuracy of the finite element methods is known to improve geometrically as the mesh size is decreased, resulting in a useful error criterion for the method. A disadvantage of the proposed methods is that no such error criterion has been developed. The inability to predict error estimates has always been a major drawback of synthesis techniques. However, the use of such methods, and use of the proposed methods, has been shown to be justified through proper physical insight and experience.

6.3. Recommendations for Future Work

Obviously the next step is the application of these proposed methods to two-dimensional diffusion problems. However, the one-dimensional problem still contains areas which may deserve closer attention. One such area is the examination of the matrix properties of both the finite element method and the proposed approximation methods which are necessary in order to guarantee convergence to a positive eigenvalue and an everywhere positive flux solution. Another area is the development of error criteria for the proposed methods. The close similarity between the proposed methods and the finite element methods may allow an extension or generalization of characteristics which hitherto have belonged only to the finite element methods.

The usefulness of the proposed methods depends on their applicability and accuracy in two- and three-dimensional diffusion problems. Just as the finite element approximations can be derived in two- or three-dimensions using variational modal-nodal techniques, so can the proposed methods for multidimensional problems. The proposed trial functions could be defined as continuous at mesh nodes, but may in general be discontinuous along mesh line interfaces. In order that the flux and current trial functions not be allowed to be discontinuous at identical spatial points, the current trial functions would then have to be defined as continuous across these interfaces. The use of the proposed class of trial function forms in the two-dimensional problem will raise the challenge of extending the spatial overlapping synthesis methods of this type to multidimensional reactor problems.

REFERENCES

1. T. J. Thompson and J. G. Beckerley, The Technology of Nuclear Reactor Safety, Volume 1, M.I.T. Press (1964).

2. A. M. Weinberg and E. P. Wigner, The Physical Theory of Neutron Chain Reactors, University of Chicago Press (1958).

3. J. R. Lamarsh, Introduction to Nuclear Reactor Theory, Addison-Wesley (1966).

4. F. B. Hildebrand, Finite Difference Equations and Simulations, Prentice-Hall (1968).

5. S. Kaplan, "Synthesis Methods in Reactor Analysis," in Advances in Nuclear Science and Technology, Volume 3, Academic Press, 233 (1966).

6. P. M. Morse and H. Feshbach, Methods of Theoretical Physics, Chap. 6, McGraw-Hill (1953).

7. E. L. Wachspress, Iterative Solution of Elliptic Systems and Applications to the Neutron Diffusion Equations of Reactor Physics, Prentice-Hall (1966).

8. M. Clark, Jr. and K. F. Hansen, Numerical Methods of Reactor Analysis, Academic Press (1964).

9. R. Avery, "Theory of Coupled Reactors," Proc. 2nd U. N. Intern. Conf. Peaceful Uses of Atomic Energy, Geneva, 12, 182 (1958).

10. W. H. Köhler, "Summary of Derivations of Coupled Point Reactor Kinetics Equations," Proc. National Topical Meeting on Coupled Reactor Kinetics, Texas A and M Univ., 192 (1967).

11. M. W. Dyes and K. F. Hansen, "Approximate Methods for Analyzing Coupled Core Kinetics," WARD-35 (1968).

12. M. Komata, "On the Derivation of Avery's Coupled Reactor Kinetics Equations," Nucl. Sci. Eng., 38, 193 (1969).

13. J. B. Yasinsky and A. F. Henry, "Some Numerical Experiments Concerning Space-Time Reactor Kinetics Behavior," Nucl. Sci. Eng., 22, 171 (1965).

14. P. T. Antonopoulos, "Large Mesh Model Development Study," Sc.D. Thesis, M.I.T., April, 1972.

15. M. G. Steveson, "Investigations on the Macroscopic Nodal Approach to Space-Dependent Nuclear Reactor Kinetics," Ph.D. Thesis, University of Texas at Austin (1968).

16. E. L. Wachspress, "Digital Computations of Space-Time Variations of Neutron Fluxes," KAPL-2090 (1960).

17. W. M. Stacey, Jr., Modal Approximations: Theory and an Application to Reactor Physics, Research Monograph No. 41, M.I.T. Press (1967).

18. A. Foderaco and H. L. Garabedian, "Two-Group Reactor Kinetics," Nucl. Sci. Eng., 14, 22 (1962).

19. D. R. Harris, S. Kaplan, and S. G. Marbolis, "Modal Analysis of Flux Tilt: Transients in a Nonuniform Reactor," Trans. Am. Nucl. Soc., 2 (2), 178 (1959).

20. A. F. Henry, "The Application of Inhour Modes to the Description of Nonseparable Reactor Transients," Nucl. Sci. Eng., 13, 22 (1964).

21. E. L. Wachspress, "Variational Methods and Neutron Flux Synthesis," CONF-690401, 271 (1969).

22. E. L. Wachspress, R. D. Burgess, and S. Baron, "Multichannel Flux Synthesis," Nucl. Sci. Eng., 12, 381 (1962).

23. A. J. Buslik, "Interface Conditions for Few Group Neutron Diffusion Problems with Flux-Adjoint Weighted Constants," Nucl. Sci. Eng., 32, 233 (1968).

24. W. B. Terney, "Interface Conditions for Discontinuous Flux Synthesis Methods," Nucl. Sci. Eng., 41, 303 (1970).

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

26. J. B. Yasinsky and L. R. Foulke, "Improved Spatial Differencing for the Solution of Space-Time Problems Arising from the Movement of Reactor Materials," Nucl. Sci. Eng., 44, 72 (1971).

27. W. M. Stacey, Jr., "Variational Flux Synthesis Methods for Multigroup Neutron Diffusion Theory," Nucl. Sci. Eng., 47, 449 (1972).

28. M. Becker, The Principles and Applications of Variational Methods, Research Monograph No. 27, M.I.T. Press (1964).

29. A. F. Henry, "Variational Techniques as a Means of Deriving Approximation Methods," in M.I.T. Course 22.25 Notes, Advanced Reactor Physics, Chap. 15, p. 104 (Spring, 1969).

30. V. Luco and P. Lambropoulos, "Functionals for Discontinuous Trial Function Flux Synthesis," J. Nucl. Energy A/B, 24, p. 551 (1970).

31. G. C. Pomraning, "A Variational Description of Dissipative Processes," J. of Nuclear Energy A/B, Vol. 20, p. 617 (1966).

32. C. M. Kang and K. F. Hansen, "Finite Element Methods for the Neutron Diffusion Equations," Trans. Am. Nucl. Soc., Vol. 14, No. 1, p. 199 (1971).

33. C. M. Kang and K. F. Hansen, "Finite Element Methods for Space-Time Reactor Analysis," Sc.D. Thesis, Dept. of Nucl. Eng., M.I.T., MITNE-135 (1971).

34. B. W. Roos and W. C. Sangren, "Some Aspects of the Use of Digital Computers in Nuclear Reactor Design," in Advances in Nuclear Science and Technology, Vol. 2, p. 303 (1964).

35. A. F. Henry, "Demonstration That Homogenized Constants for a Cell Depend on Properties of the Surrounding Medium," in M.I.T. Course 22.25 Notes, Advanced Reactor Physics, Chap. 12, p. 63 (Spring, 1969).

36. G. Strang, "The Finite Element Method and Approximation Theory," Numerical Solution of Partial Differential Equations – II, SYNSPADE 1970, Ed. B. Hubbard, p. 547, Academic Press (1971).

37. G. Fix and G. Strang, "Fourier Analysis of the Finite Element Method in Ritz-Galerkin Theory," Studies in Applied Math., 48, 265 (1969).

38. G. Birkhoff, C. deBoor, B. Swartz, and B. Wendroff, "Rayleigh-Ritz Approximation by Piecewise Cubic Polynomials," J. SIAM Numer. Anal., Vol. 3, No. 2, p. 188 (1966).

39. J. J. Goël, "Construction of Basic Functions for Numerical Utilization of Ritz's Method," Numer. Math., 12, p. 435 (1968).

40. A. Radkowsky, Editor, Naval Reactors Physics Handbook, Vol. 1, Selected Basic Techniques, "Homogenization Techniques," Chap. 4.3, USAEC, p. 620 (1964).

41. R. E. Alcouffe and R. W. Albrecht, "A Generalization of the Finite Difference Approximation Method with an Application to Space-Time Nuclear Reactor Kinetics," Nucl. Sci. Eng., 39, 1 (1970).

42. S. Nakamura, "A Variational Rebalancing Method for Linear Iterative Convergence Schemes of Neutron Diffusion and Transport Equations," Nucl. Sci. Eng., 39, 278 (1970).

43. S. Nakamura, "Coarse Mesh Acceleration of Iterative Solution of Neutron Diffusion Equations," Nucl. Sci. Eng., 43, 116 (1971).

44. A. Radkowsky, Editor, Naval Reactors Physics Handbook, Vol. 1, Selected Basic Techniques, "Reactor Physics Computations," Chap. 7.2, USAEC, p. 1414 (1964).

45. G. Bilodeau and L. A. Hageman, "A Survey of Numerical Methods in the Solution of Diffusion Problems," WAPD-TM-64, (July, 1957).

46. G. I. Bell and S. Glasstone, Nuclear Reactor Theory, Van Nostrand Reinhold Company (1960).

47. W. W. Little, Jr. and R. W. Hardie, "2DB User's Manual – Revision 1," BNWL-831 REV 1, UC-32: Mathematics and Computers, Battelle Memorial Institute, Pacific Northwest Laboratories, Richland, Washington (August, 1969).

48. L. A. Hageman, "Numerical Methods and Techniques Used in the Two-Dimensional Neutron-Diffusion Program PDQ-5," WAPD-TM-364, UC-32: Mathematics and Computers, Bettis Atomic Power Company, Pittsburg, Pennsylvania (February, 1963).

49. E. Isaacson and H. B. Keller, Analysis of Numerical Methods, Wiley (1966).

50. R. S. Varga, Matrix Iterative Analysis, Prentice-Hall (1962).

51. G. Forsythe and C. B. Moler, Computer Solution of Linear Algebraic Systems, Prentice-Hall (1967).

52. 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, p. 8 (1969).

53. A. Radkowsky, Editor, Naval Reactors Physics Handbook, Vol. 1, Selected Basic Techniques, "Few-Group Approximations," Chap. 2.8, USAEC, p. 197 (1964).

54. M.I.T. Information Processing Center, "Stromberg-Carlson 4020," Application Program Series, AP-36 Revision 2, November 30, 1971.

BIOGRAPHICAL NOTE

The author is a native Californian. Born in San Francisco August 19, 1945, he was raised in Orange, and upon graduation from Orange High School in 1963, he received the Bank of America Award in Mathematics and the Orange County Industrial Education Association Award in Electronics.

He then attended the University of California at Berkeley, and supported by special as well as Regents' Scholarship awards, graduated in 1967 from the School of Engineering with a Bachelor of Science degree in Physics. He was elected to Tau Beta Pi, Phi Beta Kappa, and received a four-year letter in varsity gymnastics.

The author then enrolled at the Massachusetts Institute of Technology. Supported by an Atomic Energy Commission Special Fellowship in Nuclear Engineering, he received the Master of Science degree from the Department of Nuclear Engineering in 1969.

As a research associate in the Space-Time Kinetics Project in the same department, the author completed this work and obtained the degree of Doctor of Philosophy in 1972.

While at M.I.T. the author was active in many organizations, most notable of which was the M.I.T. Rugby Football Club.

APPENDICES

Appendix A

TABLE OF SYMBOLS

g	Energy group index which runs from the highest to the lowest energy group as $g = 1$ to G.
$\phi_g(r)$	Scalar neutron flux in energy group g (neutrons/$cm^2 \cdot$ sec).
$\underline{J}_g(r)$	Vector neutron current in energy group g (neutrons/$cm^2 \cdot$ sec).
$D_g(r)$	Diffusion coefficient for neutrons in energy group g (cm).
$\Sigma_g(r)$	Macroscopic total removal cross section in energy group g (cm^{-1}).
$\nu\Sigma_{fg}(r)$	Macroscopic fission-production cross section in energy group g (cm^{-1}).
$\Sigma_{gg'}(r)$	Macroscopic transfer cross section from energy group g' to energy group g (cm^{-1}).
χ_g	Fission spectrum yield in energy group g.
λ	The eigenvalue or criticality of the diffusion problem.
$\Phi(r)$, $\Phi^*(r)$	Scalar group flux column vector of length G and its adjoint.
$\underline{J}(r)$, $\underline{J}^*(r)$	Vector group current column vector of length G and its adjoint.

$\mathbb{A}(r)$	$G \times G$ group material removal, scattering, and production matrix.
$\mathbb{D}(r)$	$G \times G$ diagonal group diffusion coefficient matrix.
$U(r)$, $U^*(r)$	Scalar group flux and weighting flux trial function column vectors of length G.
$\underline{V}(r)$, $\underline{V}^*(r)$	Vector group current and weighting current trial function column vectors of length G.
k	One-dimensional spatial index which runs from the leftmost first region to the rightmost K-th region, as $k = 1$ to K.
z	The one-dimensional axis variable divided into K regions such that each region k is bounded by nodes z_k and z_{k+1}.
x	A dimensionless variable defined in each region k as $x = (z - z_k)/(z_{k+1} - z_k)$, such that $0 \leq x \leq 1$ as $z_k \leq z \leq z_{k+1}$.
F_k	Approximate one-dimensional group flux solution at node z_k.
G_k	Approximate one-dimensional group current solution at node z_k.
$\psi_k(z)$, $\psi_k^*(z)$	Detailed one-dimensional subassembly flux and weighting flux solutions in coarse mesh region k whose form is linear within each homogeneous subassembly interval.
$\eta_k(z)$, $\eta_k^*(z)$	Detailed one-dimensional subassembly current and weighting current solutions in coarse mesh region k whose form is <u>constant</u> within each homogeneous subassembly interval.

$\tilde{\eta}_k(z)$, $\tilde{\eta}_k^*(z)$ — Detailed one-dimensional subassembly current and weighting current solutions in coarse mesh region k whose form is <u>linear</u> within each homogeneous subassembly interval.

$\mathbb{A}$ — Discretized matrix form of the G×G group diffusion, absorption, and scattering matrices.

$\mathbb{B}$ — Discretized matrix form of the G×G group fission-production matrix.

$\underline{F}$ — The unknown approximate group flux solution vector which may contain group current unknowns.

$\%\lambda$ — Normalized eigenvalue percent error:
$\%\lambda = (\lambda_{\text{Reference}} - \lambda_{\text{Method}}) / \lambda_{\text{Reference}} \times 100\%$.

P(k) — Fractional power produced in coarse mesh region k when the total power produced has been normalized to unity.

$\%P(k)$ — Normalized fractional power percent error:
$\%P(k) = [P(k)_{\text{Reference}} - P(k)_{\text{Method}}] / P(k)_{\text{Reference}} \times 100\%$.

Appendix B

DIFFERENCE EQUATION COEFFICIENTS RESULTING FROM USE OF THE FINITE ELEMENT APPROXIMATION METHODS

The G×G matrix coefficients resulting from the conventional finite difference approximation, the linear finite element approximation, and the cubic Hermite finite element approximation in one-dimensional multigroup diffusion theory are defined below in sections B.1, B.2, and B.3, respectively. The coefficients are given in terms of assumed homogeneous regional nuclear constants through the use of the G×G group matrices $\mathbb{D}_k$ and $\boldsymbol{\Lambda}_k$, where $\boldsymbol{\Lambda}_k = \mathbb{M}_k - \mathbb{T}_k - \frac{1}{\lambda}\mathbb{B}_k$, which are defined in Chapter 2 and are constant for each region k, where k = 1 to K.

More general definitions of these coefficients may be found from the coefficients resulting from the use of the proposed approximations, given in Appendix C, by requiring that $\psi_k(z)$ be constant and $\eta_k(z)$ be zero in each region k.

B.1. <u>Coefficients of the Conventional Finite Difference Equations</u> (as defined by Eqs. 2.16)

Interior Coefficients; k = 2 to K:

$$a_k = -\mathbb{D}_{k-1}/h_{k-1}$$

$$b_k = \frac{1}{2}(\boldsymbol{\Lambda}_{k-1}h_{k-1} + \boldsymbol{\Lambda}_k h_k) + \mathbb{D}_{k-1}/h_{k-1} + \mathbb{D}_k/h_k$$

$$c_k = -\mathbb{D}_k/h_k$$

Symmetry Boundary Condition Coefficients:

$$b_1 = \frac{1}{2}\mathbf{\Lambda}_1 h_1 + \mathbb{D}_1/h_1$$

$$c_1 = -\mathbb{D}_1/h_1$$

$$a_{K+1} = -\mathbb{D}_K/h_K$$

$$b_{K+1} = \frac{1}{2}\mathbf{\Lambda}_K h_K + \mathbb{D}_K/h_K$$

B.2. Coefficients of the Linear Finite Element Method Equations (as defined by Eqs. 2.30)

Interior Coefficients; k = 2 to K:

$$a_k = \frac{1}{6}\mathbf{\Lambda}_{k-1} h_{k-1} - \mathbb{D}_{k-1}/h_{k-1}$$

$$b_k = \frac{1}{3}\left[\mathbf{\Lambda}_{k-1} h_{k-1} + \mathbf{\Lambda}_k h_k\right] + \mathbb{D}_{k-1}/h_{k-1} + \mathbb{D}_k/h_k$$

$$c_k = \frac{1}{6}\mathbf{\Lambda}_k h_k - \mathbb{D}_k/h_k$$

Symmetry Boundary Condition Coefficients:

$$b_1 = \frac{1}{3}\mathbf{\Lambda}_1 h_1 + \mathbb{D}_1/h_1$$

$$c_1 = \frac{1}{6}\mathbf{\Lambda}_1 h_1 - \mathbb{D}_1/h_1$$

$$a_{K+1} = \frac{1}{6}\mathbf{\Lambda}_K h_K - \mathbb{D}_K/h_K$$

$$b_{K+1} = \frac{1}{3}\mathbf{\Lambda}_K h_K + \mathbb{D}_K/h_K$$

B.3. Coefficients of the Cubic Hermite Finite Element Method Equations (as defined by Eq. 2.33)

Interior Coefficients; k = 2 to K:

$$a1_k = \frac{9}{70}\Lambda_{k-1}h_{k-1} - \frac{6}{5}\mathbb{D}_{k-1}/h_{k-1}$$

$$a2_k = \left(-\frac{13}{420}\Lambda_{k-1}h_{k-1}^2\mathbb{D}_{k-1}^{-1} + \frac{1}{10}\right)\theta$$

$$b1_k = \frac{13}{35}\left(\Lambda_{k-1}h_{k-1}+\Lambda_k h_k\right) + \frac{6}{5}\left(\mathbb{D}_{k-1}/h_{k-1} + \mathbb{D}_k/h_k\right)$$

$$b2_k = \frac{11}{210}\left(\Lambda_{k-1}h_{k-1}^2\mathbb{D}_{k-1}^{-1} - \Lambda_k h_k^2\mathbb{D}_k^{-1}\right)\theta$$

$$c1_k = \frac{9}{70}\Lambda_k h_k - \frac{6}{5}\mathbb{D}_k/h_k$$

$$c2_k = \left(\frac{13}{420}\Lambda_k h_k^2\mathbb{D}_k^{-1} - \frac{1}{10}\right)\theta$$

$$a3_k = \left(\frac{13}{420}\mathbb{D}_{k-1}^{-1}h_{k-1}^2\Lambda_{k-1} - \frac{1}{10}\right)\theta$$

$$a4_k = \left(-\frac{1}{140}\mathbb{D}_{k-1}^{-1}h_{k-1}^3\Lambda_{k-1}\mathbb{D}_{k-1}^{-1} - \frac{1}{30}h_{k-1}\mathbb{D}_{k-1}^{-1}\right)\theta^2$$

$$b3_k = \frac{11}{210}\left(\mathbb{D}_{k-1}^{-1}h_{k-1}^2\Lambda_{k-1} - \mathbb{D}_k^{-1}h_k^2\Lambda_k\right)\theta$$

$$b4_k = \left[\frac{1}{105}\left(\mathbb{D}_{k-1}^{-1}h_{k-1}^3\Lambda_{k-1}\mathbb{D}_{k-1}^{-1} + \mathbb{D}_k^{-1}h_k^3\Lambda_k\mathbb{D}_k^{-1}\right) + \frac{2}{15}\left(h_{k-1}\mathbb{D}_{k-1}^{-1} + h_k\mathbb{D}_k^{-1}\right)\right]\theta^2$$

$$c3_k = \left(-\frac{13}{420}\mathbb{D}_k^{-1}h_k^2\Lambda_k + \frac{1}{10}\right)\theta$$

$$c4_k = \left(-\frac{1}{140}\mathbb{D}_k^{-1}h_k^3\Lambda_k\mathbb{D}_k^{-1} - \frac{1}{30}h_k\mathbb{D}_k^{-1}\right)\theta^2$$

Zero Flux Boundary Condition Coefficients:

$$b4_1 = \left(\frac{1}{105}\mathbb{D}_1^{-1} h_1^3 \Lambda_1 \mathbb{D}_1^{-1} + \frac{2}{15} h_1 \mathbb{D}_1^{-1}\right)\theta^2$$

$$c3_1 = \left(-\frac{13}{420}\mathbb{D}_1^{-1} h_1^2 \Lambda_1 + \frac{1}{10}\right)\theta$$

$$c4_1 = \left(-\frac{1}{140}\mathbb{D}_1^{-1} h_1^3 \Lambda_1 \mathbb{D}_1^{-1} - \frac{1}{30} h_1 \mathbb{D}_1^{-1}\right)\theta^2$$

$$a3_{K+1} = \left(\frac{13}{420}\mathbb{D}_K^{-1} h_K^2 \Lambda_K - \frac{1}{10}\right)\theta$$

$$a4_{K+1} = \left(-\frac{1}{140}\mathbb{D}_K^{-1} h_K^3 \Lambda_K \mathbb{D}_K^{-1} - \frac{1}{30} h_K \mathbb{D}_K^{-1}\right)\theta^2$$

$$b4_{K+1} = \left(\frac{1}{105}\mathbb{D}_K^{-1} h_K^3 \Lambda_K \mathbb{D}_K^{-1} + \frac{2}{15} h_K \mathbb{D}_K^{-1}\right)\theta^2$$

Symmetry Boundary Condition Coefficients:

$$b1_1 = \frac{13}{35}\Lambda_1 h_1 + \frac{6}{5}\mathbb{D}_1/h_1$$

$$c1_1 = \frac{9}{70}\Lambda_1 h_1 - \frac{6}{5}\mathbb{D}_1/h_1$$

$$c2_1 = \left(\frac{13}{420}\Lambda_1 h_1^2 \mathbb{D}_1^{-1} - \frac{1}{10}\right)\theta$$

$$a1_{K+1} = \frac{9}{70}\Lambda_K h_K - \frac{6}{5}\mathbb{D}_K/h_K$$

$$a2_{K+1} = \left(-\frac{13}{420}\Lambda_K h_K^2 \mathbb{D}_K^{-1} + \frac{1}{10}\right)\theta$$

$$b1_{K+1} = \frac{13}{35}\Lambda_K h_K + \frac{6}{5}\mathbb{D}_K/h_K$$

Appendix C

DIFFERENCE EQUATION COEFFICIENTS RESULTING FROM USE OF THE PROPOSED APPROXIMATION METHODS

The G×G matrix coefficients resulting from the proposed approximation methods using (1) linear basis functions, and (2) cubic Hermite basis functions, in one-dimensional multigroup diffusion theory are defined below in sections C. 1 and C. 2, respectively.

The coefficients are given as integrands of functions of x where the integration of every coefficient-integrand over a region

$$\text{Coefficient} = \int_0^1 [\text{Coefficient-Integrand}(x)]\, dx$$

is understood.

In order to simplify the forms of the coefficient-integrands, it is convenient to define the following G×G matrices:

$$\mathbb{K}_k(x) = \psi_k^{*T}(x)\, \mathbf{\Lambda}_k(x)\, h_k\, \psi_k(x)$$

$$\mathbb{L}_k(x) = \eta_k^{*T}(x)\, \mathbb{D}_k^{-1}(x)\, h_k\, \eta_k(x)$$

$$\mathbb{P}_k(x) = \psi_k^{*T}(x)\, \eta_k(x)$$

$$\mathbb{Q}_k(x) = \eta_k^{*T}(x)\, \psi_k(x)$$

$$\mathbb{R}_k(x) = \frac{1}{h_k}\, \psi_k^{*T}(x)\, \mathbb{D}_k(x)\, \psi_k(x)$$

for each region k. In each approximation below, two sets of polynomial functions $p_1(x) \ldots p_{2N}(x)$ and $q_1(x) \ldots q_{2N}(x)$ are given which represent the basis functions of the approximation and their negative derivatives, where N = 1 for the linear basis function approximations and N = 2 for the cubic Hermite basis function approximations.

The G×G coefficient-integrands are then listed in terms of these matrices and polynomials by the G×G collapsed matrices $\mathbb{E}_k^{i,j}(x)$ defined as

$$\mathbb{E}_k^{i,j}(x) = p_i(x)p_j(x)\mathbb{K}_k(x) - p_i(x)p_j(x)\mathbb{L}_k(x) + q_i(x)p_j(x)\mathbb{P}_k(x)$$
$$- p_i(x)q_j(x)\mathbb{Q}_k(x) + q_i(x)q_j(x)\mathbb{R}_k(x)$$

for given values of i and j for each region k, where k = 1 to K. It should be noted that $\mathbb{E}_k^{i,j}(x)$ is not symmetric about i and j; i.e.:

$$\mathbb{E}_k^{i,j}(x) \neq \mathbb{E}_k^{j,i}(x) , \quad \text{for } i \neq j .$$

C.1. Coefficient-Integrands of the Proposed Approximation Method Equations Using Linear Basis Functions
(as defined by Eq. 3.6)

These coefficients are given in terms of the polynomial functions

$$p_1(x) = (1-x)$$

$$p_2(x) = x$$

$$q_1(x) = 1$$

$$q_2(x) = -1$$

for use in the $\mathbb{E}_k^{i,j}(x)$ below.

Interior Coefficient-Integrands; k = 2 to K:

$$a_k(x) = \psi_{k-1}^{\overset{-1}{*T}}(1)\,\mathbb{E}_{k-1}^{2,1}(x)\,\psi_{k-1}^{-1}(0)$$

$$b_k(x) = \psi_{k-1}^{\overset{-1}{*T}}(1)\,\mathbb{E}_{k-1}^{2,2}(x)\,\psi_{k-1}^{-1}(1) + \psi_k^{\overset{-1}{*T}}(0)\,\mathbb{E}_k^{1,1}(x)\,\psi_k^{-1}(0)$$

$$c_k(x) = \psi_k^{\overset{-1}{*T}}(0)\,\mathbb{E}_k^{1,2}(x)\,\psi_k^{-1}(1)$$

Symmetry Coefficient-Integrands:

$$b_1(x) = \psi_1^{\overset{-1}{*T}}(0)\,\mathbb{E}_1^{1,1}(x)\,\psi_1^{-1}(0)$$

$$c_1(x) = c_k(x)\,; \quad \text{where } k = 1$$

$$a_{K+1}(x) = a_k(x)\,; \quad \text{where } k = K+1$$

$$b_{K+1}(x) = \psi_K^{\overset{-1}{*T}}(1)\,\mathbb{E}_K^{2,2}(x)\,\psi_K^{-1}(1)$$

Implied Zero Flux Boundary Condition Coefficient-Integrands (Corresponding with the modified trial functions of the type in Eqs. 3.9)

$$b_2(x) = \psi_1^{\overset{-1}{*T}}(1)\,[\mathbb{K}_1(x) - \mathbb{L}_1(x)]\,\psi_1^{-1}(1) + \psi_2^{\overset{-1}{*T}}(0)\,\mathbb{E}_2^{1,1}(x)\,\psi_2^{-1}(0)$$

$$b_K(x) = \psi_{K-1}^{\overset{-1}{*T}}(1)\,\mathbb{E}_{K-1}^{2,2}(x)\,\psi_{K-1}^{-1}(1) + \psi_K^{\overset{-1}{*T}}(0)\,[\mathbb{K}_K - \mathbb{L}_K]\,\psi_K^{-1}(0)$$

C.2. Coefficient-Integrands of the Proposed Approximation Method Equations Using Cubic Hermite Basis Functions (as defined in Eq. 3.16)

These coefficients are given in terms of the polynomials $p_1(x)$ through $p_4(x)$ and $q_1(x)$ through $q_4(x)$, previously defined in Eqs. 3.11 and 3.12 , for use in the $\mathbb{E}_k^{i,j}(x)$ below.

Interior Coefficient-Integrands; k = 2 to K:

$$a1_k(x) = \psi_{k-1}^{*T^{-1}}(1)\,\mathbb{E}_{k-1}^{2,1}(x)\,\psi_{k-1}^{-1}(0)$$

$$a2_k(x) = \psi_{k-1}^{*T^{-1}}(1)\,\mathbb{E}_{k-1}^{2,3}(x)\,\psi_{k-1}^{-1}(0)\,\mathbb{D}_{k-1}^{-1}(0)\,\theta$$

$$b1_k(x) = \psi_{k-1}^{*T^{-1}}(1)\,\mathbb{E}_{k-1}^{2,2}(x)\,\psi_{k-1}^{-1}(1) + \psi_{k}^{*T^{-1}}(0)\,\mathbb{E}_{k}^{1,1}(x)\,\psi_{k}^{-1}(0)$$

$$b2_k(x) = \psi_{k-1}^{*T^{-1}}(1)\,\mathbb{E}_{k-1}^{2,4}(x)\,\psi_{k-1}^{-1}(1)\,\mathbb{D}_{k-1}^{-1}(1)\,\theta$$
$$+\ \psi_{k}^{*T^{-1}}(0)\,\mathbb{E}_{k}^{1,3}(x)\,\psi_{k}^{-1}(0)\,\mathbb{D}_{k}^{-1}(0)\,\theta$$

$$c1_k(x) = \psi_{k}^{*T^{-1}}(0)\,\mathbb{E}_{k}^{1,2}(x)\,\psi_{k}^{-1}(1)$$

$$c2_k(x) = \psi_{k}^{*T^{-1}}(0)\,\mathbb{E}_{k}^{1,4}(x)\,\psi_{k}^{-1}(1)\,\mathbb{D}_{k}^{-1}(1)\,\theta$$

$$a3_k(x) = \psi_{k-1}^{*T^{-1}}(1)\,\mathbb{D}_{k-1}^{-1}(1)\,\mathbb{E}_{k-1}^{4,1}(x)\,\psi_{k-1}^{-1}(0)\,\theta$$

$$a4_k(x) = \psi_{k-1}^{*T^{-1}}(1)\,\mathbb{D}_{k-1}^{-1}(1)\,\mathbb{E}_{k-1}^{4,3}(x)\,\psi_{k-1}^{-1}(0)\,\mathbb{D}_{k-1}^{-1}(0)\,\theta^2$$

$$b3_k(x) = \psi_{k-1}^{*T^{-1}}(1)\,\mathbb{D}_{k-1}^{-1}(1)\,\mathbb{E}_{k-1}^{4,2}(x)\,\psi_{k-1}^{-1}(1)\,\theta$$

$$+\ \psi_{k}^{*T^{-1}}(0)\,\mathbb{D}_{k}^{-1}(0)\,\mathbb{E}_{k}^{3,1}(x)\,\psi_{k}^{-1}(0)\,\theta$$

$$b4_k(x) = \psi_{k-1}^{*T^{-1}}(1)\,\mathbb{D}_{k-1}^{-1}(1)\,\mathbb{E}_{k-1}^{4,4}(x)\,\psi_{k-1}^{-1}(1)\,\mathbb{D}_{k-1}^{-1}(1)\,\theta^2$$

$$+\ \psi_{k}^{*T^{-1}}(0)\,\mathbb{D}_{k}^{-1}(0)\,\mathbb{E}_{k}^{3,3}(x)\,\psi_{k}^{-1}(0)\,\mathbb{D}_{k}^{-1}(0)\,\theta^2$$

$$c3_k(x) = \psi_{k}^{*T^{-1}}(0)\,\mathbb{D}_{k}^{-1}(0)\,\mathbb{E}_{k}^{3,2}(x)\,\psi_{k}^{-1}(1)\,\theta$$

$$c4_k(x) = \psi_{k}^{*T^{-1}}(0)\,\mathbb{D}_{k}^{-1}(0)\,\mathbb{E}_{k}^{3,4}(x)\,\psi_{k}^{-1}(1)\,\mathbb{D}_{k}^{-1}(1)\,\theta^2$$

Zero Flux Boundary Condition Coefficient-Integrands:

$$b4_1(x) = \psi_{1}^{*T^{-1}}(0)\,\mathbb{D}_{1}^{-1}(0)\,\mathbb{E}_{1}^{3,3}(x)\,\psi_{1}^{-1}(0)\,\mathbb{D}_{1}^{-1}(0)\,\theta^2$$

$$c3_1(x) = c3_k(x)\,;\quad \text{where } k = 1$$

$$c4_1(x) = c4_k(x)\,;\quad \text{where } k = 1$$

$$a3_{K+1}(x) = a3_k(x)\,;\quad \text{where } k = K+1$$

$$a4_{K+1}(x) = a4_{K+1}(x)\,;\quad \text{where } k = K+1$$

$$b4_{K+1}(x) = \psi_{K}^{*T^{-1}}(1)\,\mathbb{D}_{K}^{-1}(1)\,\mathbb{E}_{K}^{4,4}(x)\,\psi_{K}^{-1}(1)\,\mathbb{D}_{K}^{-1}(1)\,\theta^2$$

Symmetry Boundary Condition Coefficient-Integrands:

$$b1_1(x) = \overset{-1}{\psi_1^{*T}}(0)\,\mathbb{E}_1^{1,1}(x)\,\psi_1^{-1}(0)$$

$$c1_1(x) = c1_k(x)\,; \quad \text{where } k = 1$$

$$c2_1(x) = c2_k(x)\,; \quad \text{where } k = 1$$

$$a1_{K+1}(x) = a1_k(x)\,; \quad \text{where } k = K+1$$

$$a2_{K+1}(x) = a2_k(x)\,; \quad \text{where } k = K+1$$

$$b1_{K+1}(x) = \overset{-1}{\psi_K^{*T}}(1)\,\mathbb{E}_K^{2,2}(x)\,\psi_K^{-1}(1)$$

Implied Zero Flux Boundary Condition Coefficient-Integrands (corresponding with the modified trial functions of the type in Eq. 3.17):

$$\left.\begin{array}{l} c3_1(x) = c3_k(x)\,; \quad \text{where } k=1 \\ a2_2(x) = a2_k(x)\,; \quad \text{where } k=2 \\ b1_2(x) = b1_k(x)\,; \quad \text{where } k=2 \\ b2_2(x) = b2_k(x)\,; \quad \text{where } k=2 \\ b3_2(x) = b3_k(x)\,; \quad \text{where } k=2 \end{array}\right\} \text{ and where } \left\{\begin{array}{l} p_1(x) = 0 \\ p_2(x) = 1 \\ q_1(x) = 0 \\ q_2(x) = 0 \end{array}\right.$$

$$\left.\begin{array}{l} b1_K(x) = b1_k(x)\,; \quad \text{where } k=K \\ b2_K(x) = b2_k(x)\,; \quad \text{where } k=K \\ c2_K(x) = c2_k(x)\,; \quad \text{where } k=K \\ b3_K(x) = b3_k(x)\,; \quad \text{where } k=K \\ a3_{K+1}(x) = a3_k(x)\,; \quad \text{where } k=K+1 \end{array}\right\} \text{ and where } \left\{\begin{array}{l} p_1(x) = 1 \\ p_2(x) = 0 \\ q_1(x) = 0 \\ q_2(x) = 0 \end{array}\right.$$

Appendix D
DESCRIPTION OF THE COMPUTER PROGRAMS

The computer programs REF2G, LINEAR, CUBIC, and ANALYZE are described respectively in the following four sections. The programs are written in FORTRAN IV, allow double precision calculations, and were used with the I.B.M. 360/65 and 370/155 FORTRAN G compilers at the M.I.T. Information Processing Center. Sample storage requirements and execution times of the programs are summarized in Table D.1.

The power method employed in the first three programs allows a maximum of 300 iterations to converge, and program execution continues after this limit. Initial group flux shapes are sinusoidal or flat, depending upon the boundary conditions chosen.

The input and output data of each program are divided into data blocks for ease of representation as described below.

D.1. Description of Program REF2G

REF2G finds the reference solutions of the one-dimensional, two-group diffusion equations of each case study, or the detailed subassembly solutions of each subassembly, using the linear finite element approximation method. The program allows up to a total of two hundred homogeneous fine mesh regions and employs combinations of both zero flux and symmetry boundary conditions. Identical material regions can be automatically repeated with no additional input.

Options for plotting graphically the history of the converging spatial flux as well as the converging eigenvalue are also available. In addition, the program allows the calculation of the adjoint flux and current solutions.

The approximate current solutions are linear within each mesh region and are calculated from the converged flux solutions using Eqs. 4.29 and 4.31. The converged flux and current solutions and the converged adjoint solutions can be punched out for future use as described below.

A. Reference Solution Input Block

Card Type 1: Format (20A4)

An Appropriate Problem Title

Card Type 2: Format (2I5, 3E10.3, 5I5)

KR — Total number of homogeneous fine mesh regions. $KR \leq 200$.

IBC — Boundary Condition Option
1. Zero flux on both boundaries
2. Zero flux on the left, symmetry on the right
3. Symmetry on the left, zero flux on the right
4. Symmetry on both boundaries

EPS1 — Iteration tolerance to be met by differences between elements of successive iteration solution vectors:

$$|F_j^{(i)} - F_j^{(i-1)}| < \epsilon_1 ; \quad \text{for all } j$$

EPS2 — Iteration tolerance to be met by the mean square error between successive iteration solution vectors:

$$\{\sum_j [F_j^{(i)} - F_j^{(i-1)}]^2\}^{\frac{1}{2}} < \epsilon_2$$

Table D.1. Sample Storage Requirements and Execution Times of the Programs for Two-Group Results. Obtained using the M.I.T. I.B.M. 360/155.

Storage Requirements in Bytes (without overlays):

REF2G:	260 K
LINEAR:	200 K
CUBIC:	250 K
ANALYZE:	205 K

C.P.U. Execution Times in Minutes:

REF2G:	Detailed Subassembly Solutions (68 regions)[a]:	.120	
	Case 1 Reference Solution (150 regions)[b]:	.238	
	Case 4 Reference Solution (198 regions)[b]:	.644	
LINEAR:	Case 1 Synthesis (Homogenized[c]) Method (3 regions):	.284	(.157)
	Case 4 Synthesis (Homogenized[c]) Method (9 regions):	.296	(.209)
CUBIC:	Case 1 Synthesis (Homogenized[c]) Method (3 regions):	1.227	(.183)
	Case 4 Synthesis (Homogenized[c]) Method (9 regions):	1.464	(.328)
ANALYZE:	Case 1 Linear (Cubic Hermite) Basis Functions:	.087	(.108)
	Case 4 Linear (Cubic Hermite) Basis Functions:	.122	(.139)

---

a. Including adjoint flux and current calculations.

b. Not including adjoint calculations.

c. Including .126 minutes for calculation of the two-group homogenized constants.

EPS3 Iteration tolerance to be met by the difference between successive iteration eigenvalues:

$$|\lambda^{(i)} - \lambda^{(i-1)}| < \epsilon_3$$

IPLOT Allows printed graphical display of the converging flux solution:

- 0 No display
- 1 Plot only the resultant normalized flux
- 2 Plot a normalized history of the converging flux

JPLOT Allows printed graphical display of the history of the converging eigenvalue when JPLOT = 1.

IPUNCH Allows punched output when IPUNCH = 1.

ISEE Allows printing of storage information:

- 0 No information printed
- 1 Input regional properties are printed
- 2 Input regional properties as well as the Common/B5/ storage arrays and the Common/B3/ power method matrices are printed.

NOADJ Adjoint calculations are performed when NOADJ = 0, and bypassed if NOADJ = 1.

<u>Card Type 3</u>: Format (25I2)

ITF(k) The consecutive type-number of each region from left to right as k = 1 to KR. Allows for repeating identical regions with no additional input.

Card Type 3 is repeated KR/25 times (rounded off to the next highest integer).

<u>Card Type 4</u>: Format (2F10.5)

CHI(1), CHI(2) The fission yields χ_1 and χ_2 for the fast and thermal groups, respectively.

* An Input Region Data Block:

Repeated for each different material region; $\max_k [ITF(k)]$ times.

Card Type 5: Format (I5)

k — The consecutive mesh region number (counting from from left to right) for identification purposes.

Card Types 6, 7: Format (3F10.5, 4E10.3, /, 30X, 3E10.3)

The geometry and nuclear constants for region k:

z(1)	Beginning spatial coordinate of region k (cm)
z(2)	Ending spatial coordinate of region k (cm)
H	Width of region k (cm)
A(1)	Fast-group macroscopic total cross section in region k (cm^{-1})
F(1)	Fast-group macroscopic production cross section, $\nu\Sigma_f$, in region k (cm^{-1})
D(1)	Fast-group diffusion coefficient in region k (cm)
S	Fast-to-thermal macroscopic scattering cross section in region k (cm^{-1})
A(2)	Thermal-group macroscopic total cross section in region k (cm^{-1})
F(2)	Thermal-group macroscopic production cross section, $\nu\Sigma_f$, in region k (cm^{-1})
D(2)	Thermal-group diffusion coefficient in region k (cm)

* End of an Input Region Data Block.

** Power Method Input Block: Optional

Card Type 8: Format (F10.5)

ω — Outer iteration overrelaxation parameter $1 \leq \omega \leq 2$. Default is $\omega = 1.25$.

Card Type 9: Format (D25.14)

$\lambda^{(0)}$ — Initial eigenvalue guess. Default is $\lambda^{(0)} = 1.0$.

Card Type 10: Format (4E20.10)

$((F(g,i), i = 1 \text{ to } N), g = 1 \text{ to } 2)$ Initial group flux solution guess without zero flux boundary values. Default is $\underline{F} = 1.0$.

** End of Power Method Input Block.

B. Reference Solution Output Block

When IPUNCH = 1, REF2G punches out the number of fine mesh regions, KR, under Format (I5) followed by the converged flux solutions $\psi(g,k)$ and corresponding current solutions $\tilde{\eta}(g,k)$ for each group g and spatial node k including boundary conditions. When adjoint calculations are included, the results are punched out under Format (4D20.10) as

$$((\psi(g,k), \tilde{\eta}(g,k), \psi^*(g,k), \tilde{\eta}^*(g,k),\ k = 1 \text{ to } KR+1),\ g = 1 \text{ to } 2)$$

where the notation denotes case reference solutions as well as detailed subassembly solutions. When the adjoint calculations have been bypassed, the results are punched out under Format (2D20.10) as

$$((\psi(g,k), \tilde{\eta}(g,k),\ k = 1 \text{ to } KR+1),\ g = 1 \text{ to } 2)$$

A total of 2 KR + 3 cards are punched out.

D.2. Description of Program LINEAR

Program LINEAR forms and solves the difference equations resulting from the proposed approximation method using the linear basis functions. The program allows up to twenty-five coarse mesh regions, each of which is allowed to be broken into not more than one hundred homogeneous intervals. Combinations of both zero flux and symmetry boundary conditions as well as use of the modified trial function forms in the boundary regions are allowed. Spatial flux and eigenvalue iteration history plots are also available.

The program allows a choice of the type of weighting, Galerkin or adjoint, to be used in the approximation. Also, either form of the detailed subassembly current solutions $\eta_k(x)$ or $\tilde{\eta}_k(x)$ is allowed. In addition, identical coarse mesh regions with identical detailed subassembly solutions can be repeated implicitly.

LINEAR also calculates results of the linear finite element method when suitable input is used. Such results can be obtained by using homogenized coarse mesh region nuclear constants and defining the detailed group flux solutions to be constant and the detailed currents to be zero (or by setting ITC = 0).

Punched results using detailed subassembly solutions constitute a <u>Synthesis Method Output Block</u>, while punched output resulting from the reduction to the finite element method with homogenized regional constants constitutes a <u>Homogenized Method Output Block</u>.

A. Homogenized or Synthesis Method Input Block

Undefined input parameters are identical to those previously defined in the REF2G input.

<u>Card Type 1</u>: Format (20A4)

An Appropriate Problem Title

<u>Card Type 2</u>: Format (2I5,3E10.3,6I5)

KR Total number of coarse mesh regions. KR $\leq$ 25.

IBC	1 - 4	As previously defined
	5	Modified trial function (no tilting) in the first region, symmetry on the right
	6	Zero flux on the left, modified trial function in the last region
	7	Modified trial functions in both boundary regions

EPS1

EPS2

EPS3

IPLOT

JPLOT

IPUNCH

ISEE

ITW Type of approximation weighting desired:

0 Flux (Galerkin)

1 Adjoint

ITC Form of the detailed current solutions in all subassemblies:

0 $\eta_k(x)$, $\eta_k^*(x)$ as calculated by Fick's laws

1 $\tilde{\eta}_k(x)$, $\tilde{\eta}_k^*(x)$ as given from REF2G output

Card Type 3: Format (25I2)

ITF(k) The consecutive type-number of each coarse mesh region from left to right as k = 1 to KR. Allows for repeating identical subassemblies with no additional input.

Card Type 3 is repeated KR/25 times (rounded off to the next highest integer).

Card Type 4: Format (2F10.5)

CHI(1), CHI(2)

* An Input Subassembly Region Data Block:

Repeated for each different coarse mesh region; $\max_k [ITF(k)]$ times.

Card Type 5: Format (2I5)

k The consecutive coarse mesh region number (from left to right).

N The number of homogeneous intervals in subassembly k. $N \leq 100$.

Card Types 6, 7: Format (3F10.5, 4E10.3, /, 30X, 3E10.3)

The subassembly geometry and nuclear constants within each interval corresponding to the detailed subassembly solutions.

Repeated for each interval as i = 1 to N:

z(i) Beginning spatial coordinate of interval i (cm)

z(i+1) Ending spatial coordinate of interval i (cm)

H(i) Width of interval i (cm)

A(1, i) Fast-group macroscopic total cross section in interval i (cm^{-1})

F(1, i) Fast-group macroscopic production cross section, $\nu\Sigma_f$, in interval i (cm^{-1})

D(1, i) Fast-group diffusion coefficient in interval i (cm)

S(i) Fast-to-thermal macroscopic scattering cross section in interval i (cm^{-1})

A(2, i) Thermal-group macroscopic total cross section in interval i (cm^{-1})

F(2, i) Thermal-group macroscopic production cross section, $\nu\Sigma_f$, in interval i (cm^{-1})

D(2, i) Thermal-group diffusion coefficient in interval i (cm)

Card Type 8: Format (4D20.10)

The detailed subassembly solutions.

$((\psi(g,k), \tilde{\eta}(g,k), \psi^*(g,k), \tilde{\eta}^*(g,k),\ k = 1 \text{ to } KR+1),\ g = 1 \text{ to } 2)$

A subassembly's Reference Solution Output Block without the first card.

* END of an Input Subassembly Region Data Block.

** Expected Solution Input Block: Optional

Card Type 9: Format (D25.14)

λ_{REF} Expected eigenvalue solution. Default is $\lambda_{REF} = 1.0$.

Card Type 10: Format (4E20.10)

$((F(i,g), i = 1 \text{ to } N), g = 1 \text{ to } 2)$ Expected group flux solution without zero flux boundary values. Default is $\underline{F} = 1.0$.

** END of the Expected Solution Input Block.

*** Power Method Input Block: Optional

As previously defined in the REF2G input.

*** END of the Power Method Input Block.

B. Homogenized or Synthesis Method Output Block

When IPUNCH = 1, LINEAR punches out the total number of coarse mesh regions, KR, under Format (I5) followed by the resultant flux solutions including boundary conditions. The flux solutions are punched out under Format (2E20.7) as

$$(F(1,k), F(2,k),\ k = 1 \text{ to } KR+1)$$

These cards represent either a Homogenized or Synthesis Method Output Block, depending upon the type and form of input data used.

D.3 Description of Program CUBIC

Program CUBIC forms and solves the difference equations resulting from the proposed approximation method using the cubic Hermite basis functions. The program is very similar in form to program LINEAR and uses similar input.

A. Homogenized or Synthesis Method Input Block

The input to CUBIC is identical to that of LINEAR except for the following:

1. The boundary condition options are restricted by $1 \leq \text{IBC} \leq 4$.

2. The normalization constant θ can be included on Card Type 4 after CHI(2) under Format (3F10.5). Default is $\theta = 1.0$.

3. Both the expected group solutions and the initial group solutions of the Expected Solution and Power Method Input Blocks, respectively, are of the form ((F(g,i), i = 1 to N), g = 1 to 2) without either zero flux or zero current (or symmetry) boundary conditions. The solution vector is made up of alternating flux and current values as described in section 3.3 of Chapter 3. Default values are flux values of unity and current values of zero.

B. Homogenized or Synthesis Method Output Block

When IPUNCH = 1, CUBIC punches out the total number of coarse mesh regions, KR, under Format (I5) followed by the resultant flux and current solutions including boundary conditions. The solutions are punched out under Format (4E20.7) as

$$(F(1,k), F(2,k), G(1,k), G(2,k),\ k = 1 \text{ to } KR+1)$$

where F(g,k) represents the flux, and G(g,k) the current solution of group g at node k.

As in the case of LINEAR, these KR+2 output cards represent either a Synthesis or Homogenized Method Output Block, depending upon the type and form of input data used.

D.4 Description of Program ANALYZE

ANALYZE compares the results of the reference solution, homogenized finite element method, and the proposed synthesis method for each case study where either linear or cubic Hermite basis functions have been used in the latter methods. For each of these three methods, the program first forms the complete detailed flux solution and then normalizes the flux distributions for each method such that their total power levels are unity. The fractional (normalized) power levels produced in each coarse mesh region are then calculated, compared, and listed. Finally, the detailed group fluxes of each method are plotted graphically relative to one another using the Stromberg-Carlson Computer Recorder, SC-4020, facility at M.I.T.[54] The graphic results for each group are normalized by the largest group-flux value such that the equivalent total power levels are preserved.

A. ANALYZE Input

The input to ANALYZE is read from five device units: 1, 2, 3, 11, 12, 13, and 5. Input and output data of the reference and approximation programs are read from the former six units while the standard input unit, 5, is reserved for SC-4020 plotting information.

The input is described by "Header Cards" and previously defined Input and Output Blocks. Header cards consist of one or more cards defined as follows:

Header Card 1: Format (4I5)

Method — Indicates the type of basis function approximation:
- 1 Linear
- 2 Cubic Hermite

NK — Total number of coarse mesh regions involved.

NR — Total number of fine mesh regions involved. NR = NK except for reference solution calculations.

NAP — Number of additional points to be plotted within each coarse mesh region. Used with the homogenized finite element method calculations. NAP < 0 denotes that the additional points are to be used in the first region (reflector) only.

Header Card 2: For use in device unit 3 input when NR ≠ NK. Format (16I5).

NRNK(k) — The number of fine mesh regions which make up each coarse mesh region k, as k = 1 to NK.

The program is dimensioned to accept up to 200 fine mesh regions (or intervals) per coarse mesh region, up to 25 coarse mesh regions, and up to a grand total of 1000 fine mesh regions in each case study.

The form of the ANALYZE input is given as follows:

Input Data for Unit 1:

Header Cards

[Homogenized Method Input Block]

Input Data for Unit 2:

Header Cards

[Synthesis Method Input Block]

Input Data for Unit 3:

Header Cards

[Reference Solution Input Block]

Input Data for Unit 11:

[Homogenized Method Output Block]

Input Data for Unit 12:

[Synthesis Method Output Block]

Input Data for Unit 13:

[Reference Solution Output Block]

Input Data for Unit 5:

No SC-4020 plots are generated if this data is omitted.

Card 1: Format (20A4)

An appropriate title written above each plotted graph.

Card 2: Format (2F10.5)

XINCH — Total width of the graph in inches including labels (limited to 7.45").

YINCH — Total height of the graph in inches including labels (limited to 7.45").

Card 3: Format (I10, F10.5)

NCELL — Total number of coarse mesh regions. NCELL $\leq$ 25. (NCELL < 0 indicates that the last region is of width $\frac{1}{2}$ WCELL.)

WCELL — Width of each coarse mesh region in cm.

Card 4: Optional. Format (I10, 7F10.5)

NLL — Number of vertical light lines to be added to the plotted graphs. NLL $\leq$ 100.

XL(i) — Spatial location (cm) of the light lines; i = 1 to 7.

Card 5: Format (8F10.5)

XL(i) — As above when NLL > 7.

Appendix E

SAMPLE INPUT AND OUTPUT DATA BLOCKS FOR PROGRAMS REF2G, LINEAR, CUBIC, AND ANALYZE

(Included in only the first six copies of this report.)

E.1. REF2G SAMPLE INPUT AND OUTPUT DATA BLOCKS

SAMPLE REF2G REFERENCE SOLUTION INPUT BLOCK:

```
CASE 1 STUDY:  THREE DIFFERENT SUBASSEMBLIES.  150 FINE MESH REFERENCE SOLUTION.
  150     4      1.E-5      1.E-5      1.E-8    1    1    1    0    1
 1 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4 5 5 5 5 5 5 5 5
 5 5 5 5 5 5 5 5 4 4 4 4 3 3 3 3 2 2 2 2 1 1 1 1 1
 6 6 6 6 6 7 7 7 7 8 8 8 8 9 9 9 910101010101010 10
10101010101010 10 9 9 9 9 8 8 8 8 7 7 7 7 6 6 6 6 6
11111111111212121213131313141414141515151515151515
1515151515151515151414141413131313121212121111111111
 1.0       0.0
    1    1
 0.0       1.0        1.0        2.59  D-2 4.85  D-3 1.396 D 0 1.79  D-2
                                 5.32  D-2 6.36  D-2 3.88  D-1
    6    1
 0.0       0.5        0.5        2.59  D-2 4.85  D-3 1.396 D 0 1.79  D-2
                                 5.32  D-2 6.36  D-2 3.88  D-1
   10    1
 0.0       0.25       0.25       2.59  D-2 4.85  D-3 1.396 D 0 1.79  D-2
                                 5.32  D-2 6.36  D-2 3.88  D-1
   14    1
 0.0       0.125      0.125      2.59  D-2 4.85  D-3 1.396 D 0 1.79  D-2
                                 5.32  D-2 6.36  D-2 3.88  D-1
   18    1
 0.0       0.0625     0.0625     4.52  D-2 0.0   D 0 1.0   D 0 0.0   D 0
                                 9.59  D-1 0.0   D 0 1.0   D 0
   51    1
 0.0       1.0        1.0        2.60  D-2 5.53  D-3 1.397 D 0 1.72  D-2
                                 7.10  D-2 1.02  D-1 3.89  D-1
   56    1
 0.0       0.5        0.5        2.60  D-2 5.53  D-3 1.397 D 0 1.72  D-2
                                 7.10  D-2 1.02  D-1 3.89  D-1
```

```
  60      1
0.0         0.25        0.25        2.60  D-2 5.53  D-3 1.397 D 0 1.72  D-2
                                    7.10  D-2 1.02  D-1 3.89  D-1
  64      1
0.0         0.125       0.125       2.60  D-2 5.53  D-3 1.397 D 0 1.72  D-2
                                    7.10  D-2 1.02  D-1 3.89  D-1
  68      1
0.0         0.0625      0.0625      4.52  D-2 0.0   D 0 1.0   D 0 0.0   D 0
                                    9.59  D-1 0.0   D 0 1.0   D 0
 101      1
0.0         1.0         1.0         2.61  D-2 6.59  D-3 1.399 D 0 1.68  D-2
                                    8.32  D-2 1.29  D-1 3.87  D-1
 106      1
0.0         0.5         0.5         2.61  D-2 6.59  D-3 1.399 D 0 1.68  D-2
                                    8.32  D-2 1.29  D-1 3.87  D-1
 110      1
0.0         0.25        0.25        2.61  D-2 6.59  D-3 1.399 D 0 1.68  D-2
                                    8.32  D-2 1.29  D-1 3.87  D-1
 114      1
0.0         0.125       0.125       2.61  D-2 6.59  D-3 1.399 D 0 1.68  D-2
                                    8.32  D-2 1.29  D-1 3.87  D-1
 118      1
0.0         0.0625      0.0625      4.52  D-2 0.0   D 0 1.0   D 0 0.0   D 0
                                    9.59  D-1 0.0   D 0 1.0   D 0
```

SAMPLE REF2G SUBASSEMBLY SOLUTION INPUT BLOCK:

```
CELL A TWO GROJP CELL SOLUTION:  FUEL A + CRUCIFORM ROD.  TWO GROUP CONSTANTS.
   68    4      1.E-8      1.E-8      1.E-8     1    1    1    0    0
 1 2 3 3 3 3 4 4 5 5 5 5 5 5 6 6 6 6 7 7 7 7 7 7 7
 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 7 7 7 7 7 7 7
 6 6 6 6 5 5 5 5 5 5 4 4 3 3 3 3 2 1
 1.0       0.0
   1    1
 0.0       0.0625     0.0625     2.59   D-2 4.85   D-3 1.396 D 0 1.79   D-2
                                 5.32   D-2 6.36   D-2 3.88   D-1
   2    1
 0.0       0.9375     0.9375     2.59   D-2 4.85   D-3 1.396 D 0 1.79   D-2
                                 5.32   D-2 6.36   D-2 3.88   D-1
   3    1
 0.0       1.0        1.0        2.59   D-2 4.85   D-3 1.396 D 0 1.79   D-2
                                 5.32   D-2 6.36   D-2 3.88   D-1
   7    1
 0.0       0.5        0.5        2.59   D-2 4.85   D-3 1.396 D 0 1.79   D-2
                                 5.32   D-2 6.36   D-2 3.88   D-1
   9    1
 0.0       0.25       0.25       2.59   D-2 4.85   D-3 1.396 D 0 1.79   D-2
                                 5.32   D-2 6.36   D-2 3.88   D-1
  15    1
 0.0       0.125      0.125      2.59   D-2 4.85   D-3 1.396 D 0 1.79   D-2
                                 5.32   D-2 6.36   D-2 3.88   D-1
  19    1
 0.0       0.0625     0.0625     2.59   D-2 4.85   D-3 1.396 D 0 1.79   D-2
                                 5.32   D-2 6.36   D-2 3.88   D-1
  27    1
 0.0       0.0625     0.0625     4.52   D-2 0.0    D 0 1.0    D 0 0.0    D 0
                                 9.59   D-1 0.0    D 0 1.0    D 0
```

SAMPLE REF2G REFERENCE SOLUTION OUTPUT BLOCK (SAMPLE SUBASSEMBLY SOLUTION):

```
68
    0.1000000D 01   0.0            D 0      0.6840883D 00   0.0            D 0
    0.9999906D 00      0.2062319D-02        0.6840819D 00     -0.1410808D-02
    0.9976040D 00      0.6351434D-02        0.6824493D 00     -0.4344942D-02

   63 ADDITIONAL FAST GROUP DATA CARDS

    0.9976040D 00     -0.6351434D-02        0.6824493D 00      0.4344942D-02
    0.9999906D 00     -0.2062319D-02        0.6840819D 00      0.1410808D-02
    0.1000000D 01   0.0            D 0      0.6840883D 00   0.0            D 0
    0.3090145D 00   0.0            D 0      0.1000000D 01   0.0            D 0
    0.3090072D 00      0.4484312D-03        0.9999762D 00     -0.1451165D-02
    0.3071103D 00      0.1418028D-02        0.9938377D 00     -0.4588871D-02

   63 ADDITIONAL THERMAL GROUP DATA CARDS

    0.3071103D 00     -0.1418028D-02        0.9938377D 00      0.4588871D-02
    0.3090072D 00     -0.4484312D-03        0.9999762D 00      0.1451165D-02
    0.3090145D 00   0.0            D 0      0.1000000D 01   0.0            D 0
```

E.2. LINEAR SAMPLE INPUT AND OUTPUT DATA BLOCKS

SAMPLE LINEAR OR CUBIC HOMOGENIZED METHOD INPUT BLOCK:

```
CASE 1 STUDY:  THREE DIFFERENT SUBASSEMBLIES.  HOMOGENIZED FINITE ELEMENT METHOD
    3    4     1.E-5     1.E-5     1.E-8    1    1    1    0    1    1
 1 2 3
 1.0       0.0
    1    1
 0.0       18.0      18.0     .268878D-1.460175D-2.137952D 1.169837D-1
                              .681283D-1.625518D-1.398086D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
    2    1
 0.0       18.0      18.0     .269849D-1.524631D-2.137948D 1.163176D-1
                              .864780D-1.100222D 0.399649D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
    3    1
 0.0       18.0      18.0     .270820D-1.625116D-2.137943D 1.159361D-1
                              .989361D-1.126682D 0.398014D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
           1.0   D 0           0.0   D 0          1.0   D 0          0.0   D 0
```

SAMPLE LINEAR METHOD OUTPUT BLOCK:

```
3
    0.2444087D 00        0.6758041D-01
    0.4060866D 00        0.9039887D-01
    0.7838067D 00        0.1339626D 00
    0.1000000D 01        0.1563839D 00
```

E.3. CUBIC SAMPLE INPUT AND OUTPUT DATA BLOCKS

SAMPLE LINEAR OR CUBIC PROPOSED SYNTHESIS METHOD INPUT BLOCK:

```
CASE 1 STUDY:  THREE DIFFERENT SUBASSEMBLIES.  SUBASSEMBLY SYNTHESIS.
    3    4     1.E-5      1.E-5      1.E-8     1     1     1     0     1     1
 1 2 3
 1.0        0.0
   1   68
  0.0        0.06250    0.06250 0.259E-01 0.485E-02 0.140E 01 0.179E-01
                                0.532E-01 0.636E-01 0.388E 00
  0.06250    1.00000    0.93750 0.259E-01 0.485E-02 0.140E 01 0.179E-01
                                0.532E-01 0.636E-01 0.388E 00

    64 ADDITIONAL CARD PAIRS OF TYPE A SUBASSEMBLY INTERVAL MATERIAL INPUT

 17.00000  17.93750    0.93750 0.259E-01 0.485E-02 0.140E 01 0.179E-01
                                0.532E-01 0.636E-01 0.388E 00
 17.93750  18.00000    0.06250 0.259E-01 0.485E-02 0.140E 01 0.179E-01
                                0.532E-01 0.636E-01 0.388E 00
     0.1000000D 01    0.0             D 0       0.6840883D 00    0.0              D 0
     0.9999906D 00       0.2062319D-02          0.6840819D 00       -0.1410808D-02
     0.9976040D 00       0.6351434D-02          0.6824493D 00       -0.4344942D-02

    63 ADDITIONAL FAST GROUP DATA CARDS

     0.9976040D 00      -0.6351434D-02          0.6824493D 00        0.4344942D-02
     0.9999906D 00      -0.2062319D-02          0.6840819D 00        0.1410808D-02
     0.1000000D 01    0.0             D 0       0.6840883D 00    0.0              D 0
     0.3090145D 00    0.0             D 0       0.1000000D 01    0.0              D 0
     0.3090072D 00       0.4484312D-03          0.9999762D 00       -0.1451165D-02
     0.3071103D 00       0.1418028D-02          0.9938377D 00       -0.4588871D-02

    63 ADDITIONAL THERMAL GROUP DATA CARDS
```

```
    0.3071103D 00      -0.1418028D-02       0.9938377D 00       0.4588871D-02
    0.3090072D 00      -0.4484312D-03       0.9999762D 00       0.1451165D-02
    0.3090145D 00   0.0             D 0     0.1000000D 01   0.0             D 0
 2   68
 0.0         0.06250    0.06250 0.260E-01 0.553E-02 0.140E 01 0.172E-01
                                0.710E-01 0.102E 00 0.389E 00
 0.06250     1.00000    0.93750 0.260E-01 0.553E-02 0.140E 01 0.172E-01
                                0.710E-01 0.102E 00 0.389E 00

   64 ADDITIONAL CARD PAIRS OF TYPE B SUBASSEMBLY INTERVAL MATERIAL INPUT

17.00000  17.93750    0.93750 0.260E-01 0.553E-02 0.140E 01 0.172E-01
                              0.710E-01 0.102E 00 0.389E 00
17.93750  18.00000    0.06250 0.260E-01 0.553E-02 0.140E 01 0.172E-01
                              0.710E-01 0.102E 00 0.389E 00
    0.1000000D 01   0.0             D 0     0.6579158D 00   0.0             D 0
    0.9999912D 00       0.1938698D-02       0.6579100D 00      -0.1275500D-02
    0.9977474D 00       0.5978934D-02       0.6564338D 00      -0.3933635D-02

   63 ADDITIONAL FAST GROUP DATA CARDS

    0.9977474D 00      -0.5978934D-02       0.6564338D 00       0.3933635D-02
    0.9999912D 00      -0.1938698D-02       0.6579100D 00       0.1275500D-02
    0.1000000D 01   0.0             D 0     0.6579158D 00   0.0             D 0
    0.2294488D 00   0.0             D 0     0.1000000D 01   0.0             D 0
    0.2294442D 00       0.2792649D-03       0.9999801D 00      -0.1217112D-02
    0.2282630D 00       0.8868016D-03       0.9948319D 00      -0.3864921D-02

   63 ADDITIONAL THERMAL GROUP DATA CARDS

    0.2282630D 00      -0.8868016D-03       0.9948319D 00       0.3864921D-02
    0.2294442D 00      -0.2792649D-03       0.9999801D 00       0.1217112D-02
    0.2294488D 00   0.0             D 0     0.1000000D 01   0.0             D 0
 3   68
 0.0         0.06250    0.06250 0.261E-01 0.659E-02 0.140E 01 0.168E-01
```

```
                                     0.832E-01 0.129E 00 0.387E 00
 0.06250    1.00000     0.93750 0.261E-01 0.659E-02 0.140E 01 0.168E-01
                                     0.832E-01 0.129E 00 0.387E 00

   64 ADDITIONAL CARD PAIRS OF TYPE C SUBASSEMBLY INTERVAL MATERIAL INPUT

17.00000  17.93750      0.93750 0.261E-01 0.659E-02 0.140E 01 0.168E-01
                                0.832E-01 0.129E 00 0.387E 00
17.93750  18.00000      0.06250 0.261E-01 0.659E-02 0.140E 01 0.168E-01
                                0.832E-01 0.129E 00 0.387E 00
     0.1000000D 01    0.0            D 0       0.6615058D 00    0.0            D 0
     0.9999916D 00        0.1837899D-02        0.6615002D 00        -0.1215781D-02
     0.9978665D 00        0.5673004D-02        0.6600944D 00        -0.3752725D-02

   63 ADDITIONAL FAST GROUP DATA CARDS

     0.9978665D 00       -0.5673004D-02        0.6600944D 00         0.3752725D-02
     0.9999916D 00       -0.1837899D-02        0.6615002D 00         0.1215781D-02
     0.1000000D 01    0.0            D 0       0.6615058D 00    0.0            D 0
     0.1937796D 00    0.0            D 0       0.1000000D 01    0.0            D 0
     0.1937762D 00        0.2081898D-03        0.9999824D 00        -0.1074364D-02
     0.1928897D 00        0.6627953D-03        0.9954074D 00        -0.3420356D-02

   63 ADDITIONAL THERMAL GROUP DATA CARDS

     0.1928897D 00       -0.6627953D-03        0.9954074D 00         0.3420356D-02
     0.1937762D 00       -0.2081898D-03        0.9999824D 00         0.1074364D-02
     0.1937796D 00    0.0            D 0       0.1000000D 01    0.0            D 0
```

SAMPLE CUBIC METHOD OUTPUT BLOCK:

```
   3
      0.2503191D 00        0.0                  0.7648905D-01        0.0
      0.4035480D 00       -0.2202874D-01        0.1062932D 00        0.5636188D-03
      0.7785649D 00       -0.2731031D-01        0.1642353D 00        0.3579210D-03
      0.1000000D 01        0.0                  0.1946404D 00        0.0
```

E.4. ANALYZE SAMPLE INPUT AND OUTPUT

SAMPLE ANALYZE INPUT (CASE 1 CUBIC METHODS RESULTS):

```
//G.FT01F001 DD *
    2    3    3   35
CASE 1 HOMOGENIZED LINEAR FINITE ELEMENT METHOD INPUT BLOCK
            .
            .
/*

//G.FT02F001 DD *
    2    3    3    0
CASE 1 LINEAR SYNTHESIS METHOD INPUT BLOCK
            .
            .
/*

//G.FT03F001 DD *
    1    3  150    0
   50   50   50
CASE 1 REFERENCE SOLUTION INPUT BLOCK
            .
            .
/*

//G.FT11F001 DD *
CASE 1 HOMOGENIZED LINEAR FINITE ELEMENT METHOD OUTPUT BLOCK
            .
            .
/*
```

```
//G.FT12F001 DD *
CASE 1 LINEAR SYNTHESIS METHOD OUTPUT BLOCK
          .
          .
/*

//G.FT13F001 DD *
CASE 1 REFERENCE SOLUTION OUTPUT BLOCK
          .
          .
/*

//G.SYSIN DD *
TWO GROUP CASE 1 CUBIC RESULTS.
 6.0       6.0
         3   18.0
         6   8.5        9.5        26.5       27.5       44.5       45.5
/*
```

ANALYZE PRINTED OUTPUT: Case 1 with Linear Basis Functions.

RESULTS OF THE INTEGRATED POWER IN EACH OF THE 3 REGIONS:

CALCULATED POWER LEVELS, AND NUMBER OF SUBREGIONS PER REGION:

REGION:	HOMOGENIZED RESULTS:		SYNTHESIZED RESULTS:		REFERENCE RESULTS:	
1	1	0.1158776E 00	68	0.1086056E 00	50	0.1024175E 00
2	1	0.2585564E 00	68	0.2447225E 00	50	0.2404295E 00
3	1	0.4313925E 00	68	0.4112111E 00	50	0.4187305E 00
TOTALS:	3	0.8058265E 00	204	0.7645392E 00	150	0.7615775E 00

FRACTIONAL POWER LEVELS:

REGION:	HOMOGENEOUS RESULTS:	SYNTHESIZED RESULTS:	REFERENCE RESULTS:
1	0.1437997E 00	0.1420537E 00	0.1344807E 00
2	0.3208586E 00	0.3200914E 00	0.3156993E 00
3	0.5353416E 00	0.5378548E 00	0.5498199E 00
TOTALS:	0.9999999E 00	0.9999999E 00	0.9999999E 00

FRACTIONAL POWER NORMALIZED PERCENT ERRORS:

REGION:	(REF-HOMO)/REF %	(REF-SYNTH)/REF %	(SYNTN-HOMO)/SYNTH %
1	-0.6929624E 01	-0.5631252E 01	-0.1229154E 01
2	-0.1634251E 01	-0.1391243E 01	-0.2396725E 00
3	0.2633273E 01	0.2176184E 01	0.4672581E 00

ANALYZE PRINTED OUTPUT: Case 1 with Linear Basis Functions.

```
EXECUTING GENERAL ANALYSIS AND FLUX PLOTTING PROGRAM:

    TITLE OF PLOTTING RUN IS:    | THREE DIFFERENT SUBASSEMBLYS PROBLEM. |

REACTOR GEOMETRY PARAMETERS:
    NCELL =      3
    WCELL =   18.00000
    XMIN =    0.0
    XMAX =   54.00000
    YMIN =    0.0
    YMAX =    1.00000
    NLL   =     6
    (XL(I),I=1,NLL) =
            8.50000    9.50000   26.50000   27.50000   44.50000   45.50000
```

Appendix F

SOURCE LISTINGS OF THE PROGRAMS

FORTRAN source listings of programs REF2G, LINEAR, CUBIC, and ANALYZE are listed in only the first six copies of this report in the following four sections.

A figure of a subroutine overlay structure precedes each listing in order to indicate the construction of each program.

F.1. SOURCE LISTING of Program REF2G

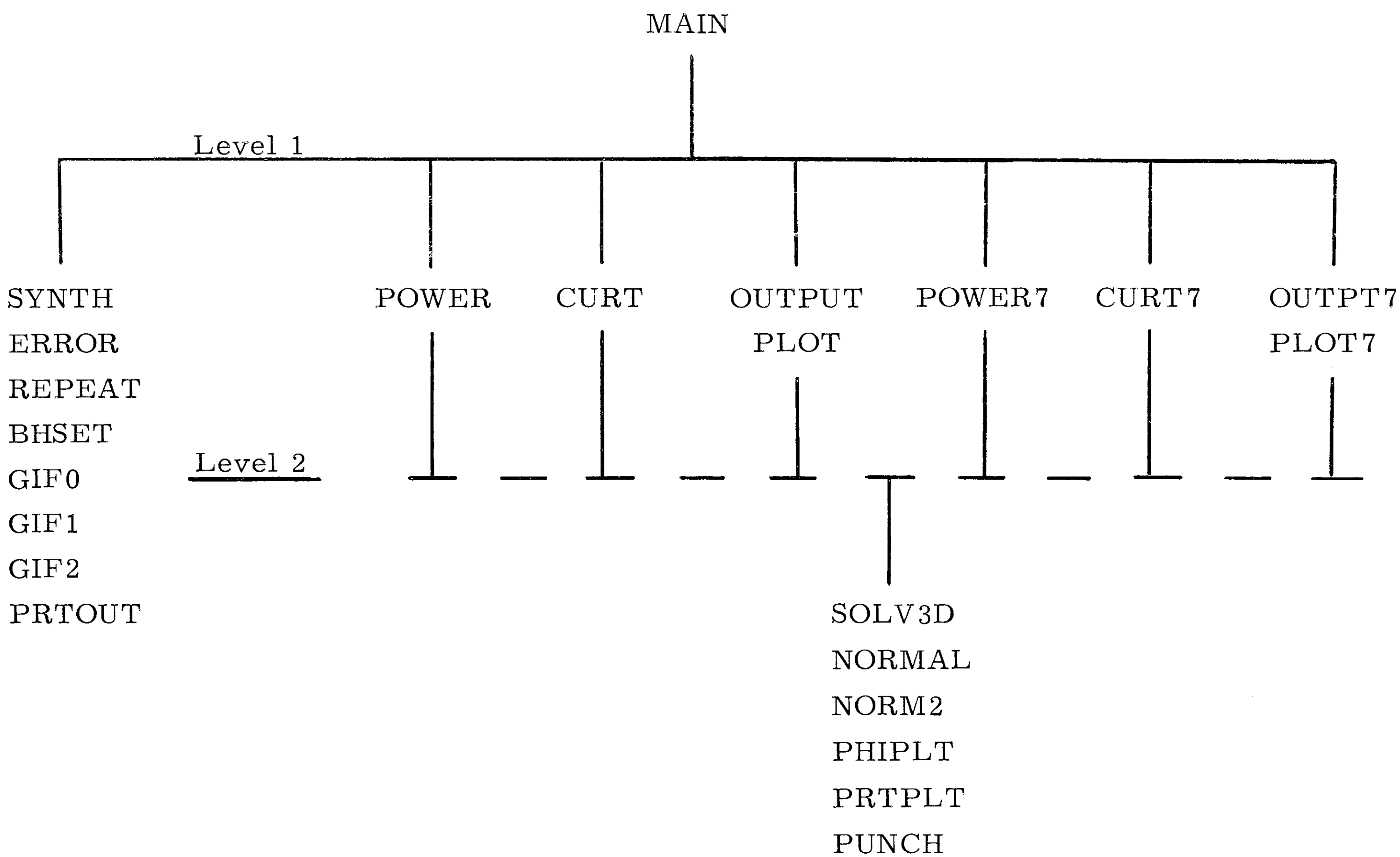


Figure F.1. Structure of Program REF2G.

```
C     PROGRAM REF2G:                                                    REF20001
C     TWO GROUP DETAILED REFERENCE AND SUBASSEMBLY SOLUTION PROGRAM.    REF20002
C                                                                       REF20003
      CALL TIMING(I1)                                                   REF20004
      CALL SYNTH                                                        REF20005
      CALL TIMING(I2)                                                   REF20006
      CALL POWER                                                        REF20007
      CALL TIMING(I3)                                                   REF20008
      CALL CURT                                                         REF20009
      CALL TIMING(I4)                                                   REF20010
      CALL OUTPUT                                                       REF20011
      CALL TIMING(I5)                                                   REF20012
      CALL POWER7                                                       REF20013
      CALL TIMING(I6)                                                   REF20014
      CALL CURT7                                                        REF20015
      CALL TIMING(I7)                                                   REF20016
      CALL OUTPT7                                                       REF20017
      CALL TIMING(I8)                                                   REF20018
C        TIMING EXECUTION                                               REF20019
      WRITE (6,30)                                                      REF20020
   30 FORMAT (1H1,'TIMING PROGRAM EXECUTION:',/)                        REF20021
      J=I2-I1                                                           REF20022
      WRITE(6,701) J                                                    REF20023
      J=I3-I2                                                           REF20024
      WRITE (6,702) J                                                   REF20025
      J=I4-I3                                                           REF20026
      WRITE (6,703) J                                                   REF20027
      J=I5-I4                                                           REF20028
      WRITE (6,704) J                                                   REF20029
      J=I6-I5                                                           REF20030
      WRITE (6,705) J                                                   REF20031
      J=I7-I6                                                           REF20032
      WRITE (6,706) J                                                   REF20033
      J=I8-I7                                                           REF20034
      WRITE(6,707) J                                                    REF20035
  701 FORMAT (1H ,' SYNTH HAS TAKEN',I6,' /100 SECONDS.')               REF20036
```

```
702 FORMAT (1H ,' POWER HAS TAKEN',I6,' /100 SECONDS.')                 REF20037
703 FORMAT (1H ,' CURT HAS TAKEN',I6,' /100 SECONDS.')                  REF20038
704 FORMAT (1H ,' OUTPUT HAS TAKEN',I5,' /100 SECONDS.')                REF20039
705 FORMAT (1H ,' POWER7 HAS TAKEN',I6,' /100 SECONDS.')                REF20040
706 FORMAT (1H ,' CURT7 HAS TAKEN',I6,' /100 SECONDS.')                 REF20041
707 FORMAT (1H ,' OUTPUT7 HAS TAKEN',I5,' /100 SECONDS.')               REF20042
    CALL TIMING(I20)                                                    REF20043
    J=I20-I1                                                            REF20044
    WRITE(6,720) J                                                      REF20045
720 FORMAT (1H0,' THIS RUN HAS TAKEN',I6,' /100 SECONDS TO RUN.')       REF20046
    STOP                                                                REF20047
    END                                                                 REF20048
```

```
      SUBROUTINE SYNTH
C        LINEAR FINITE ELEMENT METHOD:
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C        ADJOINT QUANTITIES OF VARIBLES ARE DENOTED BY 7 RATHER THAN *.
C        THUS:  PHI7  (RATHER THAN PHI*)  IS THE ADJOINT OF PHI. ETC.
      IMPLICIT REAL*8 (A-H,K-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH,ISEE,NOADJ
      COMMON /B2/ KR,NN
      COMMON /B3/ L1(201,3), L2(201,3), F1(201,3), F2(201,3), T(201,3)
      COMMON /B5/ KA0(2,200),KA1(2,200),KA2(2,200),KB0(2,200),
     X   KB1(2,200),KB2(2,200),LA0(2,200),LA1(2,200),LA2(2,200),
     X   SR0(2,200),SR1(2,200),SR2(2,200),P(2,200),P1(2,200),
     X   Q(2,200),Q1(2,200),R(2,200),P0(2,200),P07(2,200),PH(2,200),
     X   PH7(2,200),AL(2,200),BL(2,200),CL(2,200),AF(2,200),BF(2,200),
     X   CF(2,200),AT(200),BT(200),CT(200),
     X              BL0(2),   CL0(2),   BF0(2),   CF0(2),   BT0(2),
     X              CT0(2),
     X              ALK(2),   BLK(2),   AFK(2),   BFK(2),   ATK(2),
     X              BTK(2)
      COMMON /B7/ HH(200),DD(2,200)
      COMMON /CHIF/ CHI(2)
      COMMON /BH/ X(2), H(1)
      COMMON /ER/ EPS1,EPS2,EPS3
      DIMENSION PHI(2,2),PHI7(2,2),CUR(2,2),CUR7(2,2),
     X         A(2,1),F(2,1),D(2,1),S(2,1),DI(2,1),XU(2,2)
      DIMENSION ITF(200), KTF(200)
      REAL TITLE(20)
      INTEGER KR,K,KS,KS1,KR0,NN
      INTEGER NUMITF, KTF,NOADJ
      READ (5,200) TITLE
  200 FORMAT (20A4)
      WRITE (6,201) TITLE
  201 FORMAT (1H1,20A4,//)
C        READ IN THE NUMBER OF REGION TRIAL FUNCTIONS AND TYPE OF B.C.S.
C        AS WELL AS THE TOLERANCES AND THE OUTPUT TYPES DESIRED.
      READ (5,1) KR,IBC,EPS1,EPS2,EPS3,IPLOT,JPLOT,IPUNCH,ISEE,NOADJ
```

```
    1 FORMAT (2I5,3D10.3,5I5)
C         READ IN THE TYPE-NUMBER OF EACH TF REGION:
      READ (5,100) (ITF(I),I=1,KR)
  100 FORMAT (25I2)
C         READ IN THE FISSION YIELD FOR EACH GROUP:
      READ (5,101) CHI(1), CHI(2)
  101 FORMAT (2F10.5)
      KRO=KR-1
      WRITE (6,2)  KR, IBC
    2 FORMAT ('0VARIATIONAL SYNTHESIS PROGRAM #2G(200):',5X,'USING ',I3,
     X    ' SUBREACTOR REGIONS, OR TRIAL FUNCTIONS.',/,
     X    '0BOUNBRY CONDITION NUMBER (IBC) IS ',I1,'.',//,
     X    '0MATERIAL PROPERTIES AND TRIAL FUNCTIONS FOR EACH SUBREGION FO
     XLLOW:',/,
     X    '0MATERIAL PROPERTIES ARE HOMOGENEOUS IN THE INDICATED REGIONS.
     X',/,
     X    '0FLUX TRIAL FUNCTIONS ARE LINEAR IN EACH SEGMENT OF THE SUBREG
     XIONS.',/,
     X    '0CURRENT TRIAL FUNCTIONS ARE FLAT IN EACH OF THE ',
     X    'SUBREGIONS.')
      WRITE (6,20) EPS1,EPS2,EPS3,IPLOT,JPLOT,IPUNCH,ISEE,NOADJ
   20 FORMAT (//,'0TOLERANCES TO POWER ARE :  EPS1 = ',1PD10.3,/,
     X    28X,'EPS2 = ',1PD10.3,/,28X,'EPS3 = ',1PD10.3,/,
     X    '0OUTPUT PARAMETERS TO POWER ARE:  IPLOT = ',I1,/,
     X    34X,'JPLOT = ',I1,/,34X,'IPUNCH = ',I1,/,
     X    34X,'ISEE = ',I1,/,
     X    34X,'NOADJ = ',I1,'.')
      WRITE (6,22) CHI(1), CHI(2)
   22 FORMAT (/,'0FISSION YIELDS ARE:  CHI(1) =',F10.5,/,
     X    22X,'CHI(2) =',F10.5)
      IF ((KR.LE.2).AND.(IBC.EQ.1))  CALL ERROR(1,KR)
      IF (KR.GT.200)  CALL ERROR(2,KR)
      IF (EPS1.LT.1.0E-16)  CALL ERROR(6,1)
      IF (EPS2.LT.1.0E-16)  CALL ERROR(6,2)
      IF (EPS3.LT.1.0E-16)  CALL ERROR(6,3)
      IF ((IBC.LT.1).OR.(IBC.GT.4))  CALL ERROR(7,IBC)
```

```
C        DUMMY NORMAL VECTOR: XU = UNITY.  (FOR THE INTEGRATION FUNCTIONS.)      SYNT0073
      DO 21  IG=1,2                                                           SYNT0074
      DO 21  II=1,2                                                           SYNT0075
   21 XU(IG,II)=1.0                                                           SYNT0076
C        SET FLUXES TO UNITY FOR SYNTH 2G:                                    SYNT0077
      DO 25  IG=1,2                                                           SYNT0078
      DO 25  II=1,2                                                           SYNT0079
      PHI(IG,II)=1.0                                                          SYNT0080
   25 PHI7(IG,II)=1.0                                                         SYNT0081
C        COUNTER OF THE NUMBER OF TYPE-NUMBERS OF EACH TF REGION:             SYNT0082
      NUMITF=1                                                                SYNT0083
      WRITE (6,9)                                                             SYNT0084
    9 FORMAT ('1')                                                            SYNT0085
C        BEGIN TO READ IN THE TF REGION DATA AND FILL THE ARRAYS,             SYNT0086
C        DEPENDING ON THE TYPE-NUMBER OF EACH TF REGION.                      SYNT0087
      DO 50  I=1,KR                                                           SYNT0088
      IF (ITF(I).EQ.NUMITF)  GO TO 110                                        SYNT0089
C        FILL THE ARRAYS FROM OLD TF REGION TYPES:                            SYNT0090
      J=ITF(I)                                                                SYNT0091
      CALL REPEAT(I,KTF(J))                                                   SYNT0092
      GO TO 50                                                                SYNT0093
C        READ IN THE TF REGION'S DATA FOR NEW TF REGION TYPE-NUMBERS:         SYNT0094
  110 NUMITF=NUMITF+1                                                         SYNT0095
      KTF(NUMITF-1)=I                                                         SYNT0096
C        READ THE SUBREGION NUMBER AND THE NUMBER OF REGIONS IN THE SUBREGION. SYNT0097
      READ (5,1)  K                                                           SYNT0098
      KS=1                                                                    SYNT0099
      IF (KS.GT.100)  CALL ERROR(3,I)                                         SYNT0100
      KS1=KS+1                                                                SYNT0101
C        CHECK FOR IMPROPER SEQUENCING OF INPUT DATA:                         SYNT0102
      IF (I.NE.K)  CALL ERROR(4,I)                                            SYNT0103
C        READ IN THE GEOMETRY AND THE MATERIAL PROPERTIES OF THIS REGION:     SYNT0104
      READ (5,3) (X(J),X(J+1),H(J),A(1,J),F(1,J),D(1,J),S(1,J),               SYNT0105
     X                         A(2,J),F(2,J),D(2,J),J=1,KS)                  SYNT0106
    3 FORMAT (3F10.5,4E10.3,/,30X,3E10.3)                                     SYNT0107
C        WRITING OUT THE INPUT INFORMATION:                                   SYNT0108
```

```
      IF (ISEE.EQ.0)  GO TO 14
      WRITE (6,10) K,KR,KS,(J,X(J),X(J+1),H(J),A(1,J),F(1,J),D(1,J),
     X   S(1,J),A(2,J),F(2,J),D(2,J),J=1,KS)
   10 FORMAT ('0INPUT MATERIAL PROPERTIES FOR SUBREGION NUMBER ',I3,
     X   ',  OF THE ',I3,' USED.',//,
     X   5X,'THIS SUBREGION IS DIVIDED INTO ',I3,' HOMOGENEOUS SEGMENTS
     XAS FOLLOWS:',//,
     X   5X,'FAST GROUP CONSTANTS APPEAR FIRST:',//,
     X   ' REGION #',5X,'INTERNAL BOUNDARIES',13X,'WIDTH',3X,
     X   'ABSORB. CX (1/CM)',3X,'FISSION CX (1/CM)',6X,'DIFFUSION (CM)',
     X   4X,'SCATT. CX (1/CM)',/,
     X   5X,'I',11X,'X(I)',9X,'X(I+1)',11X,'H(I)',13X,'A(IG,I)',13X,
     X   'F(IG,I)',13X,'D(IG,I)',14X,'S(1,I)',//,
     X   (I6,3F15.4,4D20.8,/,51X,3D20.8))
C        END OF THE IN-OUT SECTION.
   14 CONTINUE
C        DEFINING MISC. ARRAYS FOR THE INTEGRATION FUNCTIONS:
C        LEGNTH OF THE SUBREGION: HT
      HT=X(KS1)-X(1)
      HH(K)=HT
      DD(1,K)=D(1,1)
      DD(2,K)=D(2,1)
C        INVERSE OF D ARRAYS:
      DO 13  J=1,KS
      DI(1,J)=1./D(1,J)
   13 DI(2,J)=1./D(2,J)
C        FORMATION OF THE INTEGRATION FUNCTIONS:
      CALL BHSET(KS)
C        DO FOR ALL ENERGY GROUPS:
      DO 50  IG=1,2
      KA0(IG,K)=GIF0(IG,PHI7,PHI,A,KS)
      KA1(IG,K)=GIF1(IG,PHI7,PHI,A,KS)
      KA2(IG,K)=GIF2(IG,PHI7,PHI,A,KS)
      KB0(IG,K)=GIF0(IG,PHI7,PHI,F,KS)
      KB1(IG,K)=GIF1(IG,PHI7,PHI,F,KS)
      KB2(IG,K)=GIF2(IG,PHI7,PHI,F,KS)
```

```
      R(IG,K)=GIFO(IG,PHI7,PHI,D,KS)/(HT*HT)
C        NO SCATTERING IN THE LOWEST GROUP:
      IF (IG.EQ.2)  GO TO 50
      SRO(IG,K)=GIFO(IG,PHI7,PHI,S,KS)
      SR1(IG,K)=GIF1(IG,PHI7,PHI,S,KS)
      SR2(IG,K)=GIF2(IG,PHI7,PHI,S,KS)
   50 CONTINUE
      NUMITF=NUMITF-1
      WRITE (6,51) NUMITF
   51 FORMAT ('1THERE ARE ONLY',I3,' DIFFERENT TRIAL FUNCTION REGIONS.')
      WRITE (6,52) (I,ITF(I),I=1,KR)
   52 FORMAT (/,'OTABLE OF THE TRIAL FUNCTION REGION TYPES:',//,
     X    3X,'TF REGION',4X,'REGION TYPE-NUMBER',//,
     X    (I7,12X,I7))
C        DETERMINATION OF THE B.C. OPTION PARAMETERS:
C        NN  IS THE MM AND FF MATRIX BLOCK SIZE.
      IF (IBC.EQ.1)  NN=KR-1
      IF ((IBC.EQ.2).OR.(IBC.EQ.3))  NN=KR
      IF (IBC.EQ.4)  NN=KR+1
C        FORMATION OF THE COEFFICIENT VECTORS:
C        THE INTERIOR COEFFS:
      DO 60  IG=1,2
      DO 60  K=2,KR
      J=K-1
      AL(IG,K)=KA1(IG,J)-KA2(IG,J)-R(IG,J)
      BL(IG,K)=KA2(IG,J)+R(IG,J)+KAO(IG,K)-2.*KA1(IG,K)+KA2(IG,K)
     X                               +R(IG,K)
      CL(IG,K)=KA1(IG,K)-KA2(IG,K)-R(IG,K)
      AF(IG,K)=KB1(IG,J)-KB2(IG,J)
      BF(IG,K)=KB2(IG,J)+KBO(IG,K)-2.*KB1(IG,K)+KB2(IG,K)
      CF(IG,K)=KB1(IG,K)-KB2(IG,K)
      AT(K)=SR1(1,J)-SR2(1,J)
      BT(K)=SR2(1,J)+SRO(1,K)-2.*SR1(1,K)+SR2(1,K)
      CT(K)=SR1(1,K)-SR2(1,K)
   60 CONTINUE
C        THE ZERO FLUX COEFFS:
```

```
      DO 61  IG=1,2                                                     SYNT0181
      BLO(IG)=KA0(IG,1)-2.*KA1(IG,1)+KA2(IG,1)+R(IG,1)                  SYNT0182
      BFO(IG)=KB0(IG,1)-2.*KB1(IG,1)+KB2(IG,1)                          SYNT0183
      CLO(IG)=KA1(IG,1)-KA2(IG,1)-R(IG,1)                               SYNT0184
   61 CFO(IG)=KB1(IG,1)-KB2(IG,1)                                       SYNT0185
      BTO(1)=SR0(1,1)-2.*SR1(1,1)+SR2(1,1)                              SYNT0186
      CTO(1)=SR1(1,1)-SR2(1,1)                                          SYNT0187
C        THE ZERO CURRENT COEFFS:                                       SYNT0188
      K=KR                                                              SYNT0189
      DO 62  IG=1,2                                                     SYNT0190
      ALK(IG)=KA1(IG,K)-KA2(IG,K)-R(IG,K)                               SYNT0191
      BLK(IG)=KA2(IG,K)+R(IG,K)                                         SYNT0192
      AFK(IG)=KB1(IG,K)-KB2(IG,K)                                       SYNT0193
   62 BFK(IG)=KB2(IG,K)                                                 SYNT0194
      ATK(1)=SR1(1,K)-SR2(1,K)                                          SYNT0195
      BTK(1)=SR2(1,K)                                                   SYNT0196
C        ZERO MATRICES:                                                 SYNT0197
      L1(1,1)=0.                                                        SYNT0198
      L2(1,1)=0.                                                        SYNT0199
      F1(1,1)=0.                                                        SYNT0200
      F2(1,1)=0.                                                        SYNT0201
      T (1,1)=0.                                                        SYNT0202
      L1(NN,3)=0.                                                       SYNT0203
      L2(NN,3)=0.                                                       SYNT0204
      F1(NN,3)=0.                                                       SYNT0205
      F2(NN,3)=0.                                                       SYNT0206
      T (NN,3)=0.                                                       SYNT0207
C        FILL ALL THE MATRICES FOR POWER:                               SYNT0208
      J=1                                                               SYNT0209
C        DETERMINE THE LEFT BOUNDARY CONDITIONS:                        SYNT0210
      IF (IBC.LT.3)  GO TO 67                                           SYNT0211
      L1(J,2)=BLO(1)                                                    SYNT0212
      L2(J,2)=BLO(2)                                                    SYNT0213
      F1(J,2)=BFO(1)                                                    SYNT0214
      F2(J,2)=BFO(2)                                                    SYNT0215
      T(J,2)=BTO(1)                                                     SYNT0216
```

```
      L1(J,  3)=CLO(1)                                                  SYNT0217
      L2(J,  3)=CLO(2)                                                  SYNT0218
      F1(J,  3)=CFO(1)                                                  SYNT0219
      F2(J,  3)=CFO(2)                                                  SYNT0220
      T(J,  3)=CTO(1)                                                   SYNT0221
      J=J+1                                                             SYNT0222
C        FOR ALL THE INTERIOR EQUATIONS:                                SYNT0223
   67 DO 70  K=2,KR                                                     SYNT0224
      IF (J.EQ.1)  GO TO 69                                             SYNT0225
      L1(J,  1)=AL(1,K)                                                 SYNT0226
      L2(J,  1)=AL(2,K)                                                 SYNT0227
      F1(J,  1)=AF(1,K)                                                 SYNT0228
      F2(J,  1)=AF(2,K)                                                 SYNT0229
      T(J,  1)=AT(K)                                                    SYNT0230
   69 L1(J,2)=BL(1,K)                                                   SYNT0231
      L2(J,2)=BL(2,K)                                                   SYNT0232
      F1(J,2)=BF(1,K)                                                   SYNT0233
      F2(J,2)=BF(2,K)                                                   SYNT0234
      T(J,2)=BT(K)                                                      SYNT0235
      L1(J,  3)=CL(1,K)                                                 SYNT0236
      L2(J,  3)=CL(2,K)                                                 SYNT0237
      F1(J,  3)=CF(1,K)                                                 SYNT0238
      F2(J,  3)=CF(2,K)                                                 SYNT0239
      T(J,  3)=CT(K)                                                    SYNT0240
      J=J+1                                                             SYNT0241
   70 CONTINUE                                                          SYNT0242
C        DETERMINE THE RIGHT BOUNDARY CONDITIONS:                       SYNT0243
      IF ((IBC.EQ.1).OR.(IBC.EQ.3))  GO TO 80                           SYNT0244
      L1(J,  1)=ALK(1)                                                  SYNT0245
      L2(J,  1)=ALK(2)                                                  SYNT0246
      F1(J,  1)=AFK(1)                                                  SYNT0247
      F2(J,  1)=AFK(2)                                                  SYNT0248
      T(J,  1)=ATK(1)                                                   SYNT0249
      L1(J,2)=BLK(1)                                                    SYNT0250
      L2(J,2)=BLK(2)                                                    SYNT0251
      F1(J,2)=BFK(1)                                                    SYNT0252
```

```
      F2(J,2)=BFK(2)                                                    SYNT0253
      T(J,2)=BTK(1)                                                     SYNT0254
   80 CONTINUE                                                          SYNT0255
C        PRINTS OUT THE SYNTH K ARRAYS, AND THE MATRICES GIVEN TO POWER SYNT0256
C        FOR ISEE = 2.                                                  SYNT0257
      IF (ISEE.EQ.2)  CALL PRTOUT                                       SYNT0258
      RETURN                                                            SYNT0259
      END                                                               SYNT0260
```

```
      SUBROUTINE ERROR(I,J)
C        ANNOUNCES INPUT ERRORS AND TERMINATES PROGRAM EXECUTION:
      GO TO (1,2,3,4,5,6,7,8,9),I
    1 WRITE (6,101)
      GO TO 10
    2 WRITE (6,102) J
      GO TO 10
    3 WRITE (6,103) J
      GO TO 10
    4 WRITE (6,104) J
      GO TO 10
    5 WRITE (6,105) J
      GO TO 10
    6 WRITE (6,106) J
      GO TO 10
    7 CONTINUE
    8 CONTINUE
    9 CONTINUE
   10 WRITE (6,110)
  101 FORMAT ('1MUST HAVE > 2 SUBREGIONS FOR ZERO FLUX B.C.S. INVALID.')
  102 FORMAT ('1NUMBER OF SUBREGIONS =',I3,' > 25. INVALID.')
  103 FORMAT ('1SUBREGION NUMBER',I3,' HAS > 25 SECTIONS. INVALID.')
  104 FORMAT ('1INPUT ERROR IN REGION SEQUENCING AT REGION',I5,'.')
  105 FORMAT ('1Z(I) = 0. IN REGION I =',I3,'. INVALID.')
  106 FORMAT ('1THE TOLERANCE:  EPS',I1,'  IS < 1.0E-16.  INVALID.')
  107 FORMAT ('1BOUNDRY CONDITION OPTION =',I2,' < 1 OR > 4. INVALID.')
  110 FORMAT (1H0,'PROBLEM TERMINATED.')
      CALL EXIT
      RETURN
      END
```

```
      SUBROUTINE REPEAT(K,L)
C     SETS THE /B5/ ARRAYS (K) EQUAL TO PAST STORED ARRAYS (L):
      IMPLICIT REAL*8 (A-Z)
      COMMON /B5/ KA0(2,200),KA1(2,200),KA2(2,200),KB0(2,200),
     X   KB1(2,200),KB2(2,200),LA0(2,200),LA1(2,200),LA2(2,200),
     X   SR0(2,200),SR1(2,200),SR2(2,200),P(2,200),P1(2,200),
     X   Q(2,200),Q1(2,200),R(2,200),P0(2,200),P07(2,200),PH(2,200),
     X   PH7(2,200)
      COMMON /B7/ HH(200),DD(2,200)
      INTEGER K,L,G
      DO 10  G=1,2
      KA0(G,K)=KA0(G,L)
      KA1(G,K)=KA1(G,L)
      KA2(G,K)=KA2(G,L)
      KB0(G,K)=KB0(G,L)
      KB1(G,K)=KB1(G,L)
      KB2(G,K)=KB2(G,L)
      LA0(G,K)=LA0(G,L)
      LA1(G,K)=LA1(G,L)
      LA2(G,K)=LA2(G,L)
      IF (G.EQ.2)  GO TO 5
      SR0(G,K)=SR0(G,L)
      SR1(G,K)=SR1(G,L)
      SR2(G,K)=SR2(G,L)
    5 CONTINUE
      P(G,K)=P(G,L)
      P1(G,K)=P1(G,L)
      Q(G,K)=Q(G,L)
      Q1(G,K)=Q1(G,L)
      R(G,K)=R(G,L)
      P0(G,K)=P0(G,L)
      P07(G,K)=P07(G,L)
      PH(G,K)=PH(G,L)
      PH7(G,K)=PH7(G,L)
   10 CONTINUE
      HH(K)=HH(L)
```

```
      DD(1,K)=DD(1,L)
      DD(2,K)=DD(2,L)
      RETURN
      END
```

```
      SUBROUTINE BHSET(K)
C        FIRST OF 4 ANALYTICAL INTEGRATION ROUTINES.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /BH/ X(2),H(1),H2(1),H3(1),H4(1),H5(1)
      DO 1  I=1,K
      H2(I)=X(I+1)**2-X(I)**2
      H3(I)=X(I+1)**3-X(I)**3
      H4(I)=X(I+1)**4-X(I)**4
      H5(I)=X(I+1)**5-X(I)**5
    1 CONTINUE
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION GIFO(IG,Y,Z,C,K)
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /BH/ X(2),H(1),H2(1),H3(1),H4(1),H5(1)
      DIMENSION Y(2,2), Z(2,2), C(2,1)
      SUM = 0.0
      DO 1  I=1,K
      SUM=C(IG,I)*(Y(IG,I)*Z(IG,I)*H(I)+H(I)*(Z(IG,I)*
     X    (Y(IG,I+1)-Y(IG,I))+Y(IG,I)*(Z(IG,I+1)-Z(IG,I)))/2.
     X    +H(I)*(Y(IG,I+1)-Y(IG,I))*(Z(IG,I+1)-Z(IG,I))/3.) + SUM
    1 CONTINUE
      GIFO = SUM
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION GIF1(IG,Y,Z,C,K)
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /BH/ X(2),H(1),H2(1),H3(1),H4(1),H5(1)
      DIMENSION Y(2,2), Z(2,2), C(2,1)
      SUM = 0.0
      DO 1  I=1,K
      SUM=C(IG,I)*((H2(I)/2.-X(1)*H(I))*Y(IG,I)*Z(IG,I)
     X   +(Z(IG,I)*(Y(IG,I+1)-Y(IG,I))+Y(IG,I)*
     X   (Z(IG,I+1)-Z(IG,I)))*(1./H(I))*(H3(I)/3.-H2(I)*(X(I)+X(1))/2.
     X   +X(I)*X(1)*H(I))+(Y(IG,I+1)-Y(IG,I))*(Z(IG,I+1)-Z(IG,I))
     X   *(H4(I)/4.-H3(I)*(2.*X(I)+X(1))/3.+H2(I)*(X(I)*X(I)+2.*X(I)
     X   *X(1))/2.-X(1)*X(I)*X(I)*H(I))/(H(I)*H(I))) + SUM
    1 CONTINUE
      GIF1 = SUM/(X(K+1)-X(1))
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION GIF2(IG,Y,Z,C,K)
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /BH/ X(2),H(1),H2(1),H3(1),H4(1),H5(1)
      DIMENSION Y(2,2), Z(2,2), C(2,1)
      SUM = 0.0
      DO 1  I=1,K
      SUM=C(IG,I)*(Y(IG,I)*Z(IG,I)*(H3(I)/3.-X(1)*H2(I)+X(1)*X(1)*H(I))
     X   +(1./H(I))*(Z(IG,I)*(Y(IG,I+1)-Y(IG,I))+Y(IG,I)*(Z(IG,I+1)
     X   -Z(IG,I)))*(H4(I)/4.-H3(I)*(2.*X(1)+X(I))/3.+H2(I)*(X(1)*X(1)
     X   +2.*X(1)*X(I))/2.-X(1)*X(1)*X(I)*H(I))+(1./(H(I)**2))
     X   *(Y(IG,I+1)-Y(IG,I))*(Z(IG,I+1)-Z(IG,I))*(H5(I)/5.
     X   -H4(I)*(X(1)+X(I))/2.+H3(I)*(X(1)*X(1)+4.*X(1)*X(I)+X(I)*X(I))
     X   /3.-H2(I)*(X(1)*X(I)*X(I)+X(1)*X(1)*X(I))
     X   +X(1)*X(1)*X(I)*X(I)*H(I))) + SUM
    1 CONTINUE
      GIF2 = SUM/((X(K+1)-X(1))**2)
      RETURN
      END
```

```
      SUBROUTINE PRTOUT
C     PRINTS OUT THE /B5/ ARRAYS AND THE MATRICES GIVEN TO POWER:
      IMPLICIT REAL*8 (A-H,K-Z)
      COMMON /B2/ KR, N
      COMMON /B3/ L1(201,3), L2(201,3), F1(201,3), F2(201,3), T(201,3)
      COMMON /B5/ KA0(2,200),KA1(2,200),KA2(2,200),KB0(2,200),
     X   KB1(2,200),KB2(2,200),LA0(2,200),LA1(2,200),LA2(2,200),
     X   SR0(2,200),SR1(2,200),SR2(2,200),P(2,200),P1(2,200),
     X   Q(2,200),Q1(2,200),R(2,200),P0(2,200),P07(2,200),PH(2,200),
     X   PH7(2,200)
      INTEGER KR, G, N
C         KA AND KB ARRAYS:
      WRITE (6,10)
   10 FORMAT ('1   G',4X,'I',12X,'KA0(G,I)',12X,'KA1(G,I)',12X,
     X   'KA2(G,I)',12X,'KB0(G,I)',12X,'KB1(G,I)',12X,'KB2(G,I)')
      DO 11  G=1,2
      WRITE (6,12)
   11 WRITE (6,15) (G,I,KA0(G,I),KA1(G,I),KA2(G,I),KB0(G,I),KB1(G,I),
     X               KB2(G,I),I=1,KR)
   12 FORMAT (' ')
   15 FORMAT (2I5,6D20.7)
C         LA AND SR ARRAYS:
      WRITE (6,20)
   20 FORMAT ('1   G',4X,'I',12X,'LA0(G,I)',12X,'LA1(G,I)',12X,
     X   'LA2(G,I)',12X,'SR0(G,I)',12X,'SR1(G,I)',12X,'SR2(G,I)')
      G=1
      WRITE (6,12)
      WRITE (6,15) (G,I,LA0(G,I),LA1(G,I),LA2(G,I),SR0(G,I),SR1(G,I),
     X               SR2(G,I),I=1,KR)
      G=2
      WRITE (6,12)
      WRITE (6,25) (G,I,LA0(G,I),LA1(G,I),LA2(G,I),I=1,KR)
   25 FORMAT (2I5,3D20.7)
C         P, Q, AND R ARRAYS:
      WRITE (6,30)
   30 FORMAT ('1   G',4X,'I',14X,'P(G,I)',13X,'P1(G,I)',14X,'Q(G,I)',
```

```
     X   13X,'Q1(G,I)',14X,'R(G,I)')
      DO 31  G=1,2
      WRITE (6,12)
   31 WRITE (6,35) (G,I,P(G,I),P1(G,I),Q(G,I),Q1(G,I),R(G,I),I=1,KR)
   35 FORMAT (2I5,5D20.7)
C         PO AND PH ARRAYS:
      WRITE (6,40)
   40 FORMAT ('1   G',4X,'I',13X,'PO(G,I)',12X,'PO7(G,I)',13X,'PH(G,I)',
     X   12X,'PH7(G,I)')
      DO 41  G=1,2
      WRITE (6,12)
   41 WRITE (6,45) (G,I,PO(G,I),PO7(G,I),PH(G,I),PH7(G,I),I=1,KR)
   45 FORMAT (2I5,4D20.7)
C         PRINT OUT THE /B3/ MATRICES:
      WRITE (6,50)
   50 FORMAT ('1MATRIX L1:',/)
      WRITE (6,55) ((L1(I,J),J=1,3),I=1,N)
   55 FORMAT (3E12.3, 7X,3E12.3,7X,3E12.3)
      WRITE (6,60)
   60 FORMAT ('1MATRIX L2:',/)
      WRITE (6,55) ((L2(I,J),J=1,3),I=1,N)
      WRITE (6,70)
   70 FORMAT ('1MATRIX F1:',/)
      WRITE (6,55) ((F1(I,J),J=1,3),I=1,N)
      WRITE (6,80)
   80 FORMAT ('1MATRIX F2:',/)
      WRITE (6,55) ((F2(I,J),J=1,3),I=1,N)
      WRITE (6,90)
   90 FORMAT ('1MATRIX T:',/)
      WRITE (6,55) (( T(I,J),J=1,3),I=1,N)
      RETURN
      END
```

```
      SUBROUTINE POWER
C     SOLVES THE 2*N MULTIGROUP EQUATIONS:  M*PHI = (1/LAMDA)*F*PHI
C     BY THE FISSION SOURCE POWER METHOD
C     USING SIMULTANEOUS OVERRELAXATION.
C        WHERE:  M AND F ARE DOUBLE PRECISION 2N BY 2N BLOCK MATRICES;
C        AND:    PHI IS THE 2N FLUX (FAST AND THERMAL) VECTOR.
C                  L1*PHI1 = CHI1*(F1*PHI1 + F2*PHI2)
C        -T*PHI1 + L2*PHI2 = CHI2*(F1*PHI1 + F2*PHI2)
C        METHOD FOLLOWS WACHPRESS, PAGE 83.  SOLUTION BY GROUP ITERATION.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH,ISEE
      COMMON /B2/ KR,N
      COMMON /B3/ L1(201,3), L2(201,3), F1(201,3), F2(201,3), T(201,3)
      COMMON /B4/ PHI(2,201), PSI(2,201), LAMDA, ICOUT
      COMMON /B5/ S(201), ERROR(2,201), Z(201)
      COMMON /B6/ TE1(2,5),TE2(2,5),TE3(5),IN(5)
      COMMON /B7/ HH(200)
      COMMON /CHIF/ CHI(2)
      COMMON /ER/ EPS1,EPS2,EPS3
      COMMON /T/ I1,I4
      COMMON /FSTR/ PHISTR(2,201,6)
      COMMON /ESTR/ LAMSTR(300), EFSTR(2,300), EFMSTR(2,300), ERLAM(300)
      COMMON /READ5/ R5
      DIMENSION PSI1(201), PSI2(201), SQ(2), DPHI(2), ERRMAX(2)
      INTEGER N
      R5=1.
C        DEFAULT OPTIONS FOR POWER PARAMETERS:
      ALPHA=1.25
      LAMDA=1.0
      HX=0.0
      DO 505  I=1,KR
  505 HX=HX+HH(I)
      DO 555  IG=1,2
      IF (IBC.NE.4)  GO TO 551
      DO 550  I=1,N
  550 PHI(IG,I)=1.0
```

```
      GO TO 555
  551 X=3.1415926/HX
      IF (IBC.NE.1)  X=X/2.0
      SUM1=0.0
      DO 552  K=1,KR
      SUM1=SUM1+HH(K)
  552 PHI(IG,K)=DSIN(SUM1*X)
  555 CONTINUE
C        READ IN:  OVERRELAXATION PARAMETERS   ;  ALPHA  (OUTER ITERATION)
C                  INITIAL GUESS AT EIGENVALUE;  LAMDA
C                  INITIAL NORMALIZED FLUX    ;  PHI(1-N)
      READ (5,506,END=510) ALPHA
      READ (5,502,END=510) LAMDA
      READ (5,503) (PHI(1,I),I=1,N)
      READ (5,503) (PHI(2,I),I=1,N)
  506 FORMAT (F10.5)
  502 FORMAT (E25.14)
  503 FORMAT ((4E20.10))
      GO TO 511
  510 R5=0.
  511 CONTINUE
C        STORING FOR PRINTING THE MULTIGRCUP FLUX SHAPE.
      DO 11  IG=1,2
      DO 10  I=1,N
   10 PHISTR(IG,I,2)=PHI(IG,I)
C        FILL RUNNING COORD IN PHISTR
      KR1=KR+1
      DO 11  I=1,KR1
   11 PHISTR(IG,I,1)=DFLOAT(I)
C        IK IS THE FLUX PLOTTING CCUNTER.
      IK=1
C        STORES THE ITERATION NUMBER FOR FLUX HISTORY PLOTTING:
      IN(1)=0
C        STORES TEMPORARY ERRORS FOR FLUX HISTORY PLOTTING:
      TE1(1,1)=0.
      TE1(2,1)=0.
```

```
      TE2(1,1)=0.
      TE2(2,1)=0.
      TE3(1)=0.0
C        EIGENVALUE OF THE PREVIOUS ITERATION:
      LAMB4=LAMDA
C        THE MAXIMUM NUMBER OF ALLOWED ITERATIONS: ICMAX
      ICMAX=300
C        PRINT OUT THE POWER METHOD PARAMETER INFORMATION:
      WRITE (6,700) ICMAX,ALPHA,LAMDA,(PHI(1,I),I=1,N)
      WRITE (6,701) (PHI(2,I),I=1,N)
  700 FORMAT ('1EXECUTING MULTIGROUP FISSION SOURCE POWER ITERATION METH
     XOD.',///,
     X   5X,'MAXIMUM NUMBER OF ALLOWABLE ITERATIONS:',/,
     X   10X,'ICMAX =',I4,///,
     X   5X,'OUTER ITERATION RELAXATION PARAMETER:',/,
     X   10X,'ALPHA =',F7.3,//,
     X   5X,'INITIAL GUESS AT EIGENVALUE:',/,
     X   10X,'LAMBDA =',E22.14,//,
     X   5X,'INITIAL GUESS AT THE GROUP FLUX SHAPE CONNECTION POINTS:',
     X   //,8X,'FAST GROUP:',/,
     X   10X,'F(K)''S =',4E25.14,/,(18X,4E25.14))
  701 FORMAT ('0',7X,'THERMAL GROUP:',/,
     X   10X,'F(K)''S =',4E25.14,/,(18X,4E25.14))
C        BEGIN ITERATION LOOP.
      ICOUT=0
C        ICOUT IS THE OUTER ITERATION COUNTER.
   20 ICOUT=ICOUT+1
      IF (ICOUT.GT.ICMAX)  GO TO 100
C        FORM THE ITERATION SOURCE VECTOR, S;  AND ITS L-2 NORM, SUM1:
      SUM1=0.
      DO 15  I=1,N
      S(I)=0.
      I0=1
      I1=3
      IF (I.EQ.1)  I0=2
      IF (I.EQ.N)  I1=2
```

```
      DO 14  J=I0,I1
      K=I-2+J
   14 S(I)=S(I)+F1(I,J)*PHI(1,K)+F2(I,J)*PHI(2,K)
   15 SUM1=SUM1+S(I)**2
      SUM1=DSQRT(SUM1)
      SUM1=SUM1*(CHI(1)+CHI(2))
C        SOLVE FOR THE NEW GROUP FLUX VECTORS: PSI:
C        FAST GROUP; SOURCE VECTOR:
      DO 25  I=1,N
   25 Z(I)=CHI(1)*S(I)
C        FAST FLUX:
      CALL SOLV3D(N,L1,PSI1,Z)
C        THERMAL GROUP; SOURCE VECTOR:
      DO 27  I=1,N
      Z(I)=0.
      I0=1
      I1=3
      IF (I.EQ.1)  I0=2
      IF (I.EQ.N)  I1=2
      DO 26  J=I0,I1
      K=I-2+J
   26 Z(I)=Z(I)+T(I,J)*PSI1(K)
   27 Z(I)=Z(I)+CHI(2)*S(I)
C        THERMAL FLUX:
      CALL SOLV3D(N,L2,PSI2,Z)
C        FORM NEW SOURCE VECTOR FROM THE NEW UNNORMALIZED FLUXES; PSI:
      SUM2=0.
      DO 29  I=1,N
      S(I)=0.
      I0=1
      I1=3
      IF (I.EQ.1)  I0=2
      IF (I.EQ.N)  I1=2
      DO 28  J=I0,I1
      K=I-2+J
   28 S(I)=S(1)+F1(I,J)*PSI1(K)+F2(I,J)*PSI2(K)
```

```
   29 SUM2=SUM2+S(I)**2
      SUM2=DSQRT(SUM2)
      SUM2=SUM2*(CHI(1)+CHI(2))
C         CALCULATION OF THE EIGENVALUE:
      LAMDA=SUM2/SUM1
      LAMSTR(ICOUT)=LAMDA
      ERRLAM=DABS(LAMDA-LAMB4)
C         PUT PSI1 AND PSI2 INTO BIGGER PSI:
      DO 30  I=1,N
      PSI(1,I)=PSI1(I)
   30 PSI(2,I)=PSI2(I)
C         POINT BY POINT SIMULTANEOUS RELAXATION FLUX ITERATION:
      X=ALPHA
C         DO NOT RELAX DURING THE FIRST THREE ITERATIONS:
      IF (ICOUT.LE.3)  X=1.0
C         CALCULATE THE NEW GROUP FLUX ITERATES AND GROUP ERRORS:
      DO 40  IG=1,2
      SQ(IG)=0.
      DO 40  I=1,N
      ERROR(IG,I)=PSI(IG,I)/LAMDA-PHI(IG,I)
      SQ(IG)=SQ(IG)+ERROR(IG,I)**2
      PHI(IG,I)=PHI(IG,I) + X*ERROR(IG,I)
C         AND FOR PLOTTING PURPOSES:
      PSI(IG,I)=PHI(IG,I)
   40 CONTINUE
      DO 34  IG=1,2
   34 SQ(IG)=DSQRT(SQ(IG))
C         NORMALIZE PSI:
C         NORMALIZES BOTH ARRAY GROUPS TO 1.0:
      CALL NORM2(PSI,N)
      DO 36  IG=1,2
C         ERRMAX(IG) = THE MAX ERROR BETWEEN THE GROUP ITERATION FLUXES:
      ERRMAX(IG)=ERROR(IG,1)
      DO 36  I=2,N
      IF (DABS(ERROR(IG,I)).GT.ERRMAX(IG))  ERRMAX(IG)=DABS(ERROR(IG,I))
   36 CONTINUE
```

```
      IF (IPLOT.NE.2)  GO TO 45                                          POW 0181
C        THE FOLLOWING IS FOR NICELY PLOTTING THE GROUP FLUX HISTORY.    POW 0182
         DO 41  IG=1,2                                                   POW 0183
         DO 41  I=1,N                                                    POW 0184
   41    ERROR(IG,I)=PSI(IG,I)                                           POW 0185
C           ERROR NOW CONTAINS THE NEW NORMALIZED FLUX ITERATE PHI.      POW 0186
         JK=IK                                                           POW 0187
         IF (IK.EQ.0)  JK=5                                              POW 0188
         DO 42  IG=1,2                                                   POW 0189
         DO 42  I=1,N                                                    POW 0190
         IF (DABS(ERROR(IG,I)-PHISTR(IG,I,JK+1)).GE.0.01)  GO TO 43      POW 0191
   42    CONTINUE                                                        POW 0192
C           FLUX HAS NOT CHANGED ENOUGH FOR PLOTTING.                    POW 0193
         GO TO 45                                                        POW 0194
C           SAVE THE NORMALIZED FLUX FOR PLOTTING:                       POW 0195
   43    IK=IK+1                                                         POW 0196
         IN(IK)=ICOUT                                                    POW 0197
         TE3(IK)=ERRLAM                                                  POW 0198
         DO 44  IG=1,2                                                   POW 0199
         TE1(IG,IK)=ERRMAX(IG)                                           POW 0200
         TE2(IG,IK)=SQ(IG)                                               POW 0201
         DO 44  I=1,N                                                    POW 0202
   44    PHISTR(IG,I,IK+1)=ERROR(IG,I)                                   POW 0203
         IF (IK.NE.5)  GO TO 45                                          POW 0204
C           PLOT THE LAST FIVE SAVED FLUXES:                             POW 0205
         CALL PHIPLT(5)                                                  POW 0206
         IK=0                                                            POW 0207
   45 CONTINUE                                                           POW 0208
C        ERROR CRITERIA FOR ACCEPTANCE OF CONVERGENCE.                   POW 0209
      IFLAG1=0                                                           POW 0210
      IFLAG2=0                                                           POW 0211
      IFLAG3=0                                                           POW 0212
C        STORE THE ERRORS FOR COMPARISON:                                POW 0213
C             ERROR BETWEEN ITERATION EIGENVALUES:                       POW 0214
      ERLAM(ICOUT)=ERRLAM                                                POW 0215
      DO 46  IG=1,2                                                      POW 0216
```

```
C             MAXIMUM ERROR BETWEEN ITERATION FLUXES:
      EFSTR(IG,ICOUT)=ERRMAX(IG)
C             MEAN SQUARE ERROR BETWEEN ITERATION FLUXES:
      EFMSTR(IG,ICOUT)=SQ(IG)
   46 CONTINUE
      IF ((ERRMAX(1).LT.EPS1).AND.(ERRMAX(2).LT.EPS1))   IFLAG1=1
      IF ((SQ(1).LT.EPS2).AND.(SQ(2).LT.EPS2))           IFLAG2=1
      IF (ERRLAM.LT.EPS3)  IFLAG3=1
      IFLAG4=IFLAG1*IFLAG2*IFLAG3
      IF (IFLAG4.EQ.1)  GO TO 50
C        OTHERWISE CONTINUE THE ITERATION.
      LAMB4=LAMDA
      GO TO 20
   50 CONTINUE
C        CONVERGENCE ACCOMPLISHED.
C        NORMALIZE THE CONVERGED FLUX VECTOR:
      CALL NORMAL(PHI,N)
C        PLOT ANY LEFT OVER FLUX HISTORY PLOTS:
      IF ((IPLOT.EQ.2).AND.(IK.NE.0))  CALL PHIPLT(IK)
C        BOUNDRY CONDITION INSERTICNS.
      IER=0
C        IER ALLOWS B.C. INSERTIONS FOR YES AND NO CONVERGENCE:
   55 IF (IBC.EQ.4)  GO TO 90
      IF (IBC.NE.3)  GO TO 60
      PHI(1,KR+1)=0.
      PHI(2,KR+1)=0.
      GO TO 90
   60 DO 70  I=1,N
      J=N+1-I
      PHI(1,J+1)=PHI(1,J)
   70 PHI(2,J+1)=PHI(2,J)
      PHI(1,1)=0.
      PHI(2,1)=0.
      IF (IBC.NE.1)  GO TO 90
      PHI(1,KR+1)=0.
      PHI(2,KR+1)=0.
```

```
   90 IF (IER.EQ.1)  GO TO 102
      RETURN
C        NO CONVERGENCE ACCOMPLISHED:
  100 CONTINUE
C        NORMALIZE THE UNCONVERGED FLUX:
      CALL NORMAL(PHI,N)
      ICOUT=ICOUT-1
      WRITE (6,101) ICOUT
  101 FORMAT (1H1,'POWER METHOD DID NOT CONVERGE FOR THIS CASE AFTER',
     X   I4,' ITERATIONS.',//,1X,'EXECUTION TERMINATED ')
      IER=1
      GO TO 55
  102 CONTINUE
C        FOR PRINTING OUT THE EIGENVALUE HISTORY AND THE FINAL FLUX SHAPE:
      IF (IPLOT.EQ.0)  IPLOT=1
      IF (JPLOT.EQ.0)  JPLOT=1
      RETURN
      END
```

```
      SUBROUTINE CURT
C        SOLVES FOR THE CURRENT FROM THE INPUT H(K)'S AND D(K)'S
C        USING F(K)'S FROM POWER:
C        CURRENT IS LINEAR (LEAST SQUARES - VARIATIONAL) AND PUT INTO ARRAY C.
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /B2/ KR
      COMMON /B4/ F(2,201), C(2,201)
      COMMON /B5/ T(201,3), S1(201), S2(201), C1(201), C2(201)
      COMMON /B7/ H(200), D(2,200)
C        FROM THE MATRIX PROBLEM FOR LINEAR FIT OF STEP DATA:
      M=KR
      N=KR+1
      T(1,1)=0.
      T(N,3)=0.
      T(1,2)=H(1)/3.
      T(1,3)=H(1)/6.
      T(N,  1)=H(M)/6.
      T(N,2)=H(M)/3.
      S1(1)=D(1,1)*(F(1,1)-F(1,2) )/2.
      S2(1)=D(2,1)*(F(2,1)-F(2,2) )/2.
      S1(N)=D(1,M)*(F(1,M)-F(1,M+1))/2.
      S2(N)=D(2,M)*(F(2,M)-F(2,M+1))/2.
      DO 20  I=2,M
      J=I-1
      T(I,  1)=H(J)/6.
      T(I,2)=(H(J)+H(I))/3.
      T(I,  3)=H(I)/6.
      S1(I)=(D(1,J)*(F(1,J)-F(1,I))+D(1,I)*(F(1,I)-F( 1,I+1)))/2.
      S2(I)=(D(2,J)*(F(2,J)-F(2,I))+D(2,I)*(F(2,I)-F( 2,I+1)))/2.
   20 CONTINUE
      CALL SOLV3D(N,T,C1,S1)
      CALL SOLV3D(N,T,C2,S2)
      DO 30  I=1,N
      C(1,I)=C1(I)
   30 C(2,I)=C2(I)
      RETURN
```

END

```
      SUBROUTINE OUTPUT
C        PRINTS THE RESULTS OF THE METHOD.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH,ISEE,NOADJ
      COMMON /B2/ KR,N
      COMMON /B4/ PHI(2,201), CUR(2,201), LAMDA, ICOUT
      COMMON /ER/ EPS1,EPS2,EPS3
      COMMON /ESTR/ LAMSTR(300),EFSTR(2,300),EFMSTR(2,300),ERLAM(300)
      INTEGER N, NOADJ
      KR0=KR-1
      KR1=KR+1
      WRITE (6,1)
    1 FORMAT ('1RESULTS OF THE MULTIGROUP METHOD:')
      WRITE (6,10) ICOUT
   10 FORMAT (//,' PROBLEM TERMINATED AFTER',I5,
     X        ' OUTER (POWER) ITERATIONS TO:')
      WRITE (6,20) LAMDA
   20 FORMAT (/,10X,'LAMDA = ',1PE21.14)
C        PRINT OUT EIGENVALUES.
      CALL PLOT
      WRITE (6,30)
   30 FORMAT ('1RESULTS AFTER PROBLEM TERMINATION:',/,
     X   '0NUMBER',9X,'THERMAL FLUX',4X,'THERMAL CURRENT',12X,
     X   'FAST FLUX',7X,'FAST CURRENT',/)
      WRITE (6,50) (K,PHI(2,K),CUR(2,K),PHI(1,K),CUR(1,K),K=1,KR1)
   50 FORMAT (I7,1PE21.7,0PE19.7,1PE21.7,0PE19.7)
C        PRINT OUT THE STORED ITERATION ERRORS:
      WRITE (6,110) EPS1,(EFSTR(2,I),I=1,ICOUT)
      WRITE (6,111) EPS1,(EFSTR(1,I),I=1,ICOUT)
      WRITE (6,112) EPS2,(EFMSTR(2,I),I=1,ICOUT)
      WRITE (6,113) EPS3,(EFMSTR(1,I),I=1,ICOUT)
      WRITE (6,114) EPS3,(ERLAM(I),I=1,ICOUT)
  110 FORMAT ('1MAXIMUM ERRORS BETWEEN THE THERMAL FLUX ITERATIONS:',
     X   25X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  111 FORMAT ('1MAXIMUM ERRORS BETWEEN THE FAST FLUX ITERATIONS:',
     X   25X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
```

```
  112 FORMAT ('1MEAN SQUARE ERROR  BETWEEN THE THERMAL FLUX ITERATIONS:',
     X    18X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  113 FORMAT ('1MEAN SQUARE ERROR  BETWEEN THE FAST FLUX ITERATIONS:',
     X    18X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  114 FORMAT ('1ERROR BETWEEN THE ITERATICN EIGENVALUES:',
     X    28X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
      IF (NOADJ.EQ.0)  RETURN
C        OTHERWISE ADJOINT CALCULATIONS ARE NOT EXECUTED:
      WRITE (6,120)
  120 FORMAT ('1ADJOINT CALCULATIONS HAVE BEEN BYPASSED.',//,
     X    ' PROGRAM TERMINATED.')
C        IPUNCH = 1  PUNCHES OUT THE FAST FLUX FOR SYNTH 1G INPUTS:
      IF (IPUNCH.EQ.1)  WRITE (7,124) KR
      IF (IPUNCH.EQ.1)  WRITE (7,125) (PHI(1,I),CUR(1,I),I=1,KR1)
      IF (IPUNCH.EQ.1)  WRITE (7,125) (PHI(2,I),CUR(2,I),I=1,KR1)
  124 FORMAT (I5)
  125 FORMAT (2D20.10)
      CALL EXIT
      RETURN
C
      END
```

```
      SUBROUTINE PLOT
C        PLOTS OUT THE EIGENVALUE HISTORY AS A TABLE AND A GRAPH,
C        AS WELL AS PLOTTING OUT THE FINAL MULTIGROUP FLUX SHAPES.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH
      COMMON /B2/ KR
      COMMON /B4/ PHI(2,201), PSI(2,201), LAMDA, ICOUT
      COMMON /B5/ B(300,2)
      COMMON /ESTR/ LAMSTR(300)
      DIMENSION C(201,3)
C        IN ORDER TO SAVE SOME SPACE:
      EQUIVALENCE (B(1),C(1))
      INTEGER ND
      ND=201
      WRITE (6,1) (LAMSTR(I),I=1,ICOUT)
    1 FORMAT ('0TABLE OF EIGENVALUES DURING THE POWER ITERATION:',
     X         //,(1P5E25.14))
      IF (JPLOT.EQ.0)  GO TO 20
      DO 10  I=1,ICOUT
      B(I,1)=I
   10 B(I,2)=LAMSTR(I)
      CALL PRTPLT(1,B,ICOUT,2,ICOUT,0,300,2,1)
      WRITE (6,11)
   11 FORMAT ('0PLOT OF THE EIGENVALUE HISTORY THROUGH THE ITERATIONS.')
   20 IF (IPLOT.EQ.0)  RETURN
      KR1=KR+1
      DO 30  I=1,KR1
      C(I,1)=I
      C(I,2)=PHI(1,I)
   30 C(I,3)=PHI(2,I)
      CALL PRTPLT(2,C,KR1,3,KR1,0,ND,3,2)
      WRITE (6,31)
   31 FORMAT ('0FINAL CONVERGED CONNECTING FLUX POINTS; F(K).',//,
     X   5X,'FAST FLUX:     .',/,5X,'THERMAL FLUX:  -')
      RETURN
      END
```

```
      SUBROUTINE POWER7                                                 POW70001
C     *** ADJOINT PROBLEM ***                                           POW70002
C     SOLVES THE 2*N MULTIGROUP ADJOINT EQUATIONS:                      POW70003
C             M*PHI = (1/LAMDA)*F*PHI                                   POW70004
C     BY THE FISSION SOURCE POWER METHOD                                POW70005
C     USING SIMULTANEOUS OVERRELAXATION.                                POW70006
C        WHERE:  M AND F ARE DOUBLE PRECISION 2N BY 2N BLOCK MATRICES;  POW70007
C        AND:    PHI IS THE 2N ADJOINT (FAST AND THERMAL) VECTOR.       POW70008
C        L1*PHI1 - T*PHI2 = CHI1*F1*PHI1 + CHI2*F1*PHI2                 POW70009
C                 L2*PHI2 = CHI1*F2*PHI1 + CHI2*F2*PHI2                 POW70010
      IMPLICIT REAL*8 (A-H,L-Z)                                         POW70011
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH,ISEE                           POW70012
      COMMON /B2/ KR,N                                                  POW70013
      COMMON /B3/ L1(201,3), L2(201,3), F1(201,3), F2(201,3), T(201,3)  POW70014
      COMMON /B47/ PHI(2,201), PSI(2,201), LAMDA, ICOUT                 POW70015
      COMMON /B5/ S(201), ERROR(2,201), Z(201)                          POW70016
      COMMON /B6/ TE1(2,5),TE2(2,5),TE3(5),IN(5)                        POW70017
      COMMON /B7/ HH(200)                                               POW70018
      COMMON /CHIF/ CHI(2)                                              POW70019
      COMMON /ER/ EPS1,EPS2,EPS3                                        POW70020
      COMMON /T/ I1,I4                                                  POW70021
      COMMON /FSTR/ PHISTR(2,201,6)                                     POW70022
      COMMON /ESTR/ LAMSTR(300), EFSTR(2,300), EFMSTR(2,300), ERLAM(300)POW70023
      COMMON /READ5/ R5                                                 POW70024
      DIMENSION PSI1(201), PSI2(201), SQ(2), DPHI(2), ERRMAX(2)         POW70025
      INTEGER N                                                         POW70026
C        DEFAULT OPTIONS FOR POWER PARAMETERS:                          POW70027
      ALPHA=1.25                                                        POW70028
      LAMDA=1.0                                                         POW70029
      HX=0.0                                                            POW70030
      DO 505  I=1,KR                                                    POW70031
  505 HX=HX+HH(I)                                                       POW70032
      DO 555  IG=1,2                                                    POW70033
      IF (IBC.NE.4)  GO TO 551                                          POW70034
      DO 550  I=1,N                                                     POW70035
  550 PHI(IG,I)=1.0                                                     POW70036
```

```
      GO TO 555                                                         POW70037
  551 X=3.1415926/HX                                                    POW70038
      IF (IBC.NE.1)  X=X/2.0                                            POW70039
      SUM1=0.0                                                          POW70040
      DO 552  K=1,KR                                                    POW70041
      SUM1=SUM1+HH(K)                                                   POW70042
  552 PHI(IG,K)=DSIN(SUM1*X)                                            POW70043
  555 CONTINUE                                                          POW70044
      IF (R5.EQ.0.)  GO TO 510                                          POW70045
C        READ IN:  OVERRELAXATION PARAMETERS  ;  ALPHA  (OUTER ITERATION) POW70046
C                  INITIAL GUESS AT EIGENVALUE;  LAMDA                  POW70047
C                  INITIAL NORMALIZED FLUX    ;  PHI(1-N)               POW70048
      READ (5,506,END=510) ALPHA                                        POW70049
      READ (5,502,END=510) LAMDA                                        POW70050
      READ (5,503) (PHI(1,I),I=1,N)                                     POW70051
      READ (5,503) (PHI(2,I),I=1,N)                                     POW70052
  506 FORMAT (F10.5)                                                    POW70053
  502 FORMAT (E25.14)                                                   POW70054
  503 FORMAT ((4E20.10))                                                POW70055
  510 CONTINUE                                                          POW70056
C        STORING FOR PRINTING THE MULTIGROUP FLUX SHAPE.                POW70057
      DO 11  IG=1,2                                                     POW70058
      DO 10  I=1,N                                                      POW70059
   10 PHISTR(IG,I,2)=PHI(IG,I)                                          POW70060
C        FILL RUNNING COORD IN PHISTR                                   POW70061
      KR1=KR+1                                                          POW70062
      DO 11  I=1,KR1                                                    POW70063
   11 PHISTR(IG,I,1)=DFLOAT(I)                                          POW70064
C        IK IS THE FLUX PLOTTING COUNTER.                               POW70065
      IK=1                                                              POW70066
C        STORES THE ITERATION NUMBER FOR FLUX HISTORY PLOTTING:         POW70067
      IN(1)=0                                                           POW70068
C        STORES TEMPORARY ERRORS FOR FLUX HISTORY PLOTTING:             POW70069
      TE1(1,1)=0.                                                       POW70070
      TE1(2,1)=0.                                                       POW70071
      TE2(1,1)=0.                                                       POW70072
```

```
      TE2(2,1)=0.
      TE3(1)=0.0
C        EIGENVALUE OF THE PREVIOUS ITERATION:
      LAMB4=LAMDA
C        THE MAXIMUM NUMBER OF ALLOWED ITERATIONS: ICMAX
      ICMAX=300
C        PRINT OUT THE POWER METHOD PARAMETER INFORMATION:
      WRITE (6,700) ICMAX,ALPHA,LAMDA,(PHI(1,I),I=1,N)
      WRITE (6,701) (PHI(2,I),I=1,N)
  700 FORMAT ('1EXECUTING MUTIGROUP ADJOINT FISSION SOURCE POWER ITERATI
     XON METHOD.',///,
     X    5X,'MAXIMUM NUMBER OF ALLOWABLE ITERATIONS:',/,
     X    10X,'ICMAX =',I4,///,
     X    5X,'OUTER ITERATION RELAXATION PARAMETER:',/,
     X    10X,'ALPHA =',F7.3,//,
     X    5X,'INITIAL GUESS AT ADJOINT EIGENVALUE:',/,
     X    10X,'LAMBDA =',E22.14,//,
     X    5X,'INITIAL GUESS AT THE GROUP FLUX SHAPE CONNECTION POINTS:',
     X    //,8X,'FAST ADJOINT GROUP:',/,
     X    10X,'F(K)''S =',4E25.14,/,(18X,4E25.14))
  701 FORMAT ('0',7X,'THERMAL ADJOINT GROUP:',/,
     X    10X,'F(K)''S =',4E25.14,/,(18X,4E25.14))
C        BEGIN ITERATION LOOP.
      ICOUT=0
C        ICOUT IS THE OUTER ITERATION COUNTER.
   20 ICOUT=ICOUT+1
      IF (ICOUT.GT.ICMAX)  GO TO 100
C        FORM THE GROUP TOTAL SOURCE S, AND ITS L-2 NORM SUM1:
C        AND THE THERMAL ADJOINT SOURCE VECTOR Z:
      SUM1=0.
      DO 15  I=1,N
      Z(I)=0.
      S(I)=0.
      I0=1
      I1=3
      IF (I.EQ.1)  I0=2
```

```
      IF (I.EQ.N)  I1=2                                             POW70109
      DO 14  J=I0,I1                                                POW70110
      K=I-2+J                                                       POW70111
      Z(I)=Z(I)+F2(I,J)*(CHI(1)*PHI(1,K)+CHI(2)*PHI(2,K))           POW70112
   14 S(I)=S(I)+F1(I,J)*(CHI(1)*PHI(1,K)+CHI(2)*PHI(2,K))           POW70113
      S(I)=S(I)+Z(I)                                                POW70114
   15 SUM1=SUM1+S(I)**2                                             POW70115
      SUM1=DSQRT(SUM1)                                              POW70116
C        SOLVE FOR THE NEW GROUP ADJOINT FLUX VECTORS; PSI:         POW70117
C        THERMAL ADJOINT FLUX:                                      POW70118
      CALL SOLV3D(N,L2,PSI2,Z)                                      POW70119
C        FAST ADJOINT GROUP; SOURCE VECTOR:                         POW70120
      DO 27  I=1,N                                                  POW70121
      S(I)=0.                                                       POW70122
      Z(I)=0.                                                       POW70123
      I0=1                                                          POW70124
      I1=3                                                          POW70125
      IF (I.EQ.1)  I0=2                                             POW70126
      IF (I.EQ.N)  I1=2                                             POW70127
      DO 26  J=I0,I1                                                POW70128
      K=I-2+J                                                       POW70129
      S(I)=S(I)+CHI(1)*F1(I,J)*PHI(1,K)+CHI(2)*F1(I,J)*PSI2(K)      POW70130
   26 Z(I)=Z(I)+T(I,J)*PSI2(K)                                      POW70131
   27 Z(I)=Z(I)+S(I)                                                POW70132
C        FAST ADJOINT FLUX:                                         POW70133
      CALL SOLV3D(N,L1,PSI1,Z)                                      POW70134
C        FORM NEW GROUP TOTAL SOURCE S FROM PSI'S, AND ITS L-2 NORM SUM2:  POW70135
      SUM2=0.                                                       POW70136
      DO 29  I=1,N                                                  POW70137
      S(I)=0.                                                       POW70138
      I0=1                                                          POW70139
      I1=3                                                          POW70140
      IF (I.EQ.1)  I0=2                                             POW70141
      IF (I.EQ.N)  I1=2                                             POW70142
      DO 28  J=I0,I1                                                POW70143
      K=I-2+J                                                       POW70144
```

```
   28 S(I)=S(I)+(F1(I,J)+F2(I,J))*(CHI(1)*PSI1(K)+CHI(2)*PSI2(K))           POW70145
   29 SUM2=SUM2+S(I)**2                                                     POW70146
      SUM2=DSQRT(SUM2)                                                      POW70147
C        CALCULATION OF THE EIGENVALUE:                                     POW70148
      LAMDA=SUM2/SUM1                                                       POW70149
      LAMSTR(ICOUT)=LAMDA                                                   POW70150
      ERRLAM=DABS(LAMDA-LAMB4)                                              POW70151
C        PUT PSI1 AND PSI2 INTO BIGGER PSI:                                 POW70152
      DO 30  I=1,N                                                          POW70153
      PSI(1,I)=PSI1(I)                                                      POW70154
   30 PSI(2,I)=PSI2(I)                                                      POW70155
C        POINT BY POINT SIMULTANEOUS RELAXATION FLUX ITERATION:             POW70156
      X=ALPHA                                                               POW70157
C        DO NOT RELAX DURING THE FIRST THREE ITERATIONS:                    POW70158
      IF (ICOUT.LE.3)  X=1.0                                                POW70159
C        CALCULATE THE NEW GROUP FLUX ITERATES AND GROUP ERRORS:            POW70160
      DO 40  IG=1,2                                                         POW70161
      SQ(IG)=0.                                                             POW70162
      DO 40  I=1,N                                                          POW70163
      ERROR(IG,I)=PSI(IG,I)/LAMDA-PHI(IG,I)                                 POW70164
      SQ(IG)=SQ(IG)+ERROR(IG,I)**2                                          POW70165
      PHI(IG,I)=PHI(IG,I) + X*ERROR(IG,I)                                   POW70166
C        AND FOR PLOTTING PURPOSES:                                         POW70167
      PSI(IG,I)=PHI(IG,I)                                                   POW70168
   40 CONTINUE                                                              POW70169
      DO 34  IG=1,2                                                         POW70170
   34 SQ(IG)=DSQRT(SQ(IG))                                                  POW70171
C        NORMALIZE PSI:                                                     POW70172
C        NORMALIZES BOTH ARRAY GROUPS TO 1.0:                               POW70173
      CALL NORM2(PSI,N)                                                     POW70174
      DO 36  IG=1,2                                                         POW70175
C        ERRMAX(IG) = THE MAX ERROR BETWEEN THE GROUP ITERATION FLUXES:     POW70176
      ERRMAX(IG)=ERROR(IG,1)                                                POW70177
      DO 36  I=2,N                                                          POW70178
      IF (DABS(ERROR(IG,I)).GT.ERRMAX(IG))  ERRMAX(IG)=DABS(ERROR(IG,I))    POW70179
   36 CONTINUE                                                              POW70180
```

```
      IF (IPLOT.NE.2)  GO TO 45                                         POW70181
C        THE FOLLOWING IS FOR NICELY PLOTTING THE GROUP FLUX HISTORY.   POW70182
         DO 41  IG=1,2                                                  POW70183
         DO 41  I=1,N                                                   POW70184
   41    ERROR(IG,I)=PSI(IG,I)                                          POW70185
C           ERROR NOW CONTAINS THE NEW NORMALIZED FLUX ITERATE PHI.     POW70186
         JK=IK                                                          POW70187
         IF (IK.EQ.0)  JK=5                                             POW70188
         DO 42  IG=1,2                                                  POW70189
         DO 42  I=1,N                                                   POW70190
         IF (DABS(ERROR(IG,I)-PHISTR(IG,I,JK+1)).GE.0.01)  GO TO 43     POW70191
   42    CONTINUE                                                       POW70192
C           FLUX HAS NOT CHANGED ENOUGH FOR PLOTTING.                   POW70193
         GO TO 45                                                       POW70194
C           SAVE THE NORMALIZED FLUX FOR PLOTTING:                      POW70195
   43    IK=IK+1                                                        POW70196
         IN(IK)=ICOUT                                                   POW70197
         TE3(IK)=ERRLAM                                                 POW70198
         DO 44  IG=1,2                                                  POW70199
         TE1(IG,IK)=ERRMAX(IG)                                          POW70200
         TE2(IG,IK)=SQ(IG)                                              POW70201
         DO 44  I=1,N                                                   POW70202
   44    PHISTR(IG,I,IK+1)=ERROR(IG,I)                                  POW70203
         IF (IK.NE.5)  GO TO 45                                         POW70204
C           PLOT THE LAST FIVE SAVED FLUXES:                            POW70205
         CALL PHIPLT(5)                                                 POW70206
         IK=0                                                           POW70207
   45 CONTINUE                                                          POW70208
C        ERROR CRITERIA FOR ACCEPTANCE OF CONVERGENCE.                  POW70209
      IFLAG1=0                                                          POW70210
      IFLAG2=0                                                          POW70211
      IFLAG3=0                                                          POW70212
C        STORE THE ERRORS FOR COMPARISON:                               POW70213
C             ERROR BETWEEN ITERATION EIGENVALUES:                      POW70214
      ERLAM(ICOUT)=ERRLAM                                               POW70215
      DO 46  IG=1,2                                                     POW70216
```

```
C             MAXIMUM ERROR BETWEEN ITERATION FLUXES:                        POW70217
      EFSTR(IG,ICOUT)=ERRMAX(IG)                                             POW70218
C             MEAN SQUARE ERROR BETWEEN ITERATION FLUXES:                    POW70219
      EFMSTR(IG,ICOUT)=SQ(IG)                                                POW70220
   46 CONTINUE                                                               POW70221
      IF ((ERRMAX(1).LT.EPS1).AND.(ERRMAX(2).LT.EPS1))   IFLAG1=1            POW70222
      IF ((SQ(1).LT.EPS2).AND.(SQ(2).LT.EPS2))           IFLAG2=1            POW70223
      IF (ERRLAM.LT.EPS3)  IFLAG3=1                                          POW70224
      IFLAG4=IFLAG1*IFLAG2*IFLAG3                                            POW70225
      IF (IFLAG4.EQ.1)  GO TO 50                                             POW70226
C        OTHERWISE CONTINUE THE ITERATION.                                   POW70227
      LAMB4=LAMDA                                                            POW70228
      GO TO 20                                                               POW70229
   50 CONTINUE                                                               POW70230
C        CONVERGENCE ACCOMPLISHED.                                           POW70231
C        NORMALIZE THE CONVERGED FLUX VECTOR:                                POW70232
      CALL NORMAL(PHI,N)                                                     POW70233
C        PLOT ANY LEFT OVER FLUX HISTORY PLOTS:                              POW70234
      IF ((IPLOT.EQ.2).AND.(IK.NE.0))  CALL PHIPLT(IK)                       POW70235
C        BOUNDRY CONDITION INSERTIONS.                                       POW70236
      IER=0                                                                  POW70237
C        IER ALLOWS B.C. INSERTIONS FOR YES AND NO CONVERGENCE:              POW70238
   55 IF (IBC.EQ.4)  GO TO 90                                                POW70239
      IF (IBC.NE.3)  GO TO 60                                                POW70240
      PHI(1,KR+1)=0.                                                         POW70241
      PHI(2,KR+1)=0.                                                         POW70242
      GO TO 90                                                               POW70243
   60 DO 70  I=1,N                                                           POW70244
      J=N+1-I                                                                POW70245
      PHI(1,J+1)=PHI(1,J)                                                    POW70246
   70 PHI(2,J+1)=PHI(2,J)                                                    POW70247
      PHI(1,1)=0.                                                            POW70248
      PHI(2,1)=0.                                                            POW70249
      IF (IBC.NE.1)  GO TO 90                                                POW70250
      PHI(1,KR+1)=0.                                                         POW70251
      PHI(2,KR+1)=0.                                                         POW70252
```

```
   90 IF (IER.EQ.1)  GO TO 102                                          POW70253
      RETURN                                                            POW70254
C        NO CONVERGENCE ACCOMPLISHED:                                   POW70255
  100 CONTINUE                                                          POW70256
C        NORMALIZE THE UNCONVERGED FLUX:                                POW70257
      CALL NORMAL(PHI,N)                                                POW70258
      ICOUT=ICOUT-1                                                     POW70259
      WRITE (6,101) ICOUT                                               POW70260
  101 FORMAT (1H1,'POWER METHOD DID NOT CONVERGE FOR THIS CASE AFTER',  POW70261
     X   I4,' ITERATIONS.',//,1X,'EXECUTION TERMINATED ')               POW70262
      IER=1                                                             POW70263
      GO TO 55                                                          POW70264
  102 CONTINUE                                                          POW70265
C        FOR PRINTING OUT THE EIGENVALUE HISTORY AND THE FINAL FLUX SHAPE:  POW70266
      IF (IPLOT.EQ.0)  IPLOT=1                                          POW70267
      IF (JPLOT.EQ.0)  JPLOT=1                                          POW70268
      RETURN                                                            POW70269
      END                                                               POW70270
```

```
      SUBROUTINE CURT7                                                  CUR70001
C        SOLVES FOR THE ADJOINT CURRENT FROM THE INPUT H(K)'S AND D(K)'S CUR70002
C        USING F(K)'S FROM POWER7:                                      CUR70003
C        CURRENT IS LINEAR (LEAST SQUARES - VARIATIONAL) AND PUT INTO ARRAY C.  CUR70004
      IMPLICIT REAL*8 (A-H,O-Z)                                         CUR70005
      COMMON /B2/ KR                                                    CUR70006
      COMMON /B47/ F(2,201), C(2,201)                                   CUR70007
      COMMON /B5/ T(201,3), S1(201), S2(201), C1(201), C2(201)          CUR70008
      COMMON /B7/ H(200), D(2,200)                                      CUR70009
C        FROM THE MATRIX PROBLEM FOR LINEAR FIT OF STEP DATA:           CUR70010
      M=KR                                                              CUR70011
      N=KR+1                                                            CUR70012
      T(1,1)=0.                                                         CUR70013
      T(N,3)=0.                                                         CUR70014
      T(1,2)=H(1)/3.                                                    CUR70015
      T(1,3)=H(1)/6.                                                    CUR70016
      T(N,  1)=H(M)/6.                                                  CUR70017
      T(N,2)=H(M)/3.                                                    CUR70018
      S1(1)=D(1,1)*(F(1,1)-F(1,2) )/2.                                  CUR70019
      S2(1)=D(2,1)*(F(2,1)-F(2,2) )/2.                                  CUR70020
      S1(N)=D(1,M)*(F(1,M)-F(1,M+1))/2.                                 CUR70021
      S2(N)=D(2,M)*(F(2,M)-F(2,M+1))/2.                                 CUR70022
      DO 20  I=2,M                                                      CUR70023
      J=I-1                                                             CUR70024
      T(I,  1)=H(J)/6.                                                  CUR70025
      T(I,2)=(H(J)+H(I))/3.                                             CUR70026
      T(I,  3)=H(I)/6.                                                  CUR70027
      S1(I)=(D(1,J)*(F(1,J)-F(1,I))+D(1,I)*(F(1,I)-F(1,I+1)))/2.        CUR70028
      S2(I)=(D(2,J)*(F(2,J)-F(2,I))+D(2,I)*(F(2,I)-F(2,I+1)))/2.        CUR70029
   20 CONTINUE                                                          CUR70030
      CALL SOLV3D(N,T,C1,S1)                                            CUR70031
      CALL SOLV3D(N,T,C2,S2)                                            CUR70032
      DO 30  I=1,N                                                      CUR70033
      C(1,I)=-C1(I)                                                     CUR70034
   30 C(2,I)=-C2(I)                                                     CUR70035
      RETURN                                                            CUR70036
```

END

```
      SUBROUTINE OUTPT7                                                 OUT70001
C        PRINTS THE RESULTS OF THE ADJOINT METHOD:                      OUT70002
      IMPLICIT REAL*8 (A-H,L-Z)                                         OUT70003
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH                                OUT70004
      COMMON /B2/ KR,N                                                  OUT70005
      COMMON /B47/ PHI(2,201), CUR(2,201), LAMDA, ICOUT                 OUT70006
      COMMON /ER/ EPS1,EPS2,EPS3                                        OUT70007
      COMMON /ESTR/ LAMSTR(300),EFSTR(2,300),EFMSTR(2,300),ERLAM(300)   OUT70008
      INTEGER N                                                         OUT70009
      KR0=KR-1                                                          OUT70010
      KR1=KR+1                                                          OUT70011
      WRITE (6,1)                                                       OUT70012
    1 FORMAT ('1RESULTS OF THE MULTIGROUP ADJOINT METHOD:')             OUT70013
      WRITE (6,10) ICOUT                                                OUT70014
   10 FORMAT (//,' PROBLEM TERMINATED AFTER',I5,                        OUT70015
     X          ' OUTER (POWER) ITERATIONS TO:')                        OUT70016
      WRITE (6,20) LAMDA                                                OUT70017
   20 FORMAT (/,10X,'ADJOINT LAMBDA = ',1PE21.14)                       OUT70018
C        PRINT OUT EIGENVALUES.                                         OUT70019
      CALL PLOT7                                                        OUT70020
      WRITE (6,30)                                                      OUT70021
   30 FORMAT ('1RESULTS AFTER PROBLEM TERMINATION:',/,                  OUT70022
     X   5X, 'ADJOINTS:',/,                                             OUT70023
     X   '0NUMBER',9X,'THERMAL FLUX',4X,'THERMAL CURRENT',12X,          OUT70024
     X   'FAST FLUX',7X,'FAST CURRENT',/)                               OUT70025
      WRITE (6,50) (K,PHI(2,K),CUR(2,K),PHI(1,K),CUR(1,K),K=1,KR1)      OUT70026
   50 FORMAT (I7,1PE21.7,0PE19.7,1PE21.7,0PE19.7)                       OUT70027
C        PRINT OUT THE STORED ITERATION ERRORS:                         OUT70028
      WRITE (6,110) EPS1,(EFSTR(2,I),I=1,ICOUT)                         OUT70029
      WRITE (6,111) EPS1,(EFSTR(1,I),I=1,ICOUT)                         OUT70030
      WRITE (6,112) EPS2,(EFMSTR(2,I),I=1,ICOUT)                        OUT70031
      WRITE (6,113) EPS3,(EFMSTR(1,I),I=1,ICOUT)                        OUT70032
      WRITE (6,114) EPS3,(ERLAM(I),I=1,ICOUT)                           OUT70033
  110 FORMAT ('1MAXIMUM ERRORS BETWEEN THE THERMAL FLUX ITERATIONS:',   OUT70034
     X   25X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))                OUT70035
  111 FORMAT ('1MAXIMUM ERRORS BETWEEN THE FAST FLUX ITERATIONS:',      OUT70036
```

```
     X    25X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))                       OUT70037
  112 FORMAT ('1MEAN SQUARE ERROR BETWEEN THE THERMAL FLUX ITERATIONS:',          OUT70038
     X    18X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))                       OUT70039
  113 FORMAT ('1MEAN SQUARE ERROR BETWEEN THE FAST FLUX ITERATIONS:',             OUT70040
     X    18X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))                       OUT70041
  114 FORMAT ('1ERROR BETWEEN THE ITERATION EIGENVALUES:',                        OUT70042
     X    28X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))                       OUT70043
C         CHECK FOR CALL TO PUNCH:                                              OUT70044
      IF (IPUNCH.EQ.1)  CALL PUNCH                                              OUT70045
      RETURN                                                                    OUT70046
      END                                                                       OUT70047
```

```
      SUBROUTINE PLOT7
C        PLOTS OUT THE EIGENVALUE HISTORY AS A TABLE AND A GRAPH,
C        AS WELL AS PLOTTING OUT THE FINAL MULTIGROUP FLUX SHAPES.
C     FOR THE ADJOINTS:
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH
      COMMON /B2/ KR
      COMMON /B47/ PHI(2,201), CUR(2,201), LAMDA, ICOUT
      COMMON /B5/ B(300,2)
      COMMON /ESTR/ LAMSTR(300)
      DIMENSION C(201,3)
C        IN ORDER TO SAVE SOME SPACE:
      EQUIVALENCE (B(1),C(1))
      INTEGER ND
      ND=201
      WRITE (6,1) (LAMSTR(I),I=1,ICOUT)
    1 FORMAT ('OTABLE OF EIGENVALUES DURING THE POWER ITERATION:',
     X         //,(1P5E25.14))
C
      IF (JPLOT.EQ.0)  GO TO 20
      DO 10  I=1,ICOUT
      B(I,1)=I
   10 B(I,2)=LAMSTR(I)
      CALL PRTPLT(1,B,ICOUT,2,ICOUT,0,300,2,1)
      WRITE (6,11)
   11 FORMAT ('OPLOT OF THE EIGENVALUE HISTORY THROUGH THE ITERATIONS.')
   20 IF (IPLOT.EQ.0)  RETURN
      KR1=KR+1
      DO 30  I=1,KR1
      C(I,1)=I
      C(I,2)=PHI(1,I)
   30 C(I,3)=PHI(2,I)
      CALL PRTPLT(2,C,KR1,3,KR1,0,ND,3,2)
      WRITE (6,31)
   31 FORMAT ('OFINAL CONVERGED CONNECTING FLUX POINTS; F(K).',//,
     X   5X,'FAST FLUX:     .',/,5X,'THERMAL FLUX:  -')
```

```
      RETURN                                                                PLT70037
      END                                                                   PLT70038
```

```
      SUBROUTINE SOLV3D(N,A,X,Y)
C     SOLVES THE N DOUBLE PRECISION MATRIX EQUATIONS:  A*X = Y,
C     FOR X - GIVEN THE N BY N TRIDIAGONAL MATRIX A
C     AND THE SOURCE VECTOR Y.
C        METHOD IS FORWARD ELIMINATION FOLLOWED BY BACKWARD SUBSTITUTION.
C        CF - WACHPRESS, PAGE 23.
      REAL*8 A, X, Y, H, P, D
      DIMENSION A(201,3), X(201), Y(201), H(201), P(201)
      IF (A(1,2).EQ.0.0)  GO TO 10
      H(1)=-A(1,3)/A(1,2)
      P(1)=Y(1)/A(1,2)
      DO 1  M=2,N
      D=A(M,2)+A(M,  1)*H(M-1)
      IF (D.EQ.0.0)  GO TO 20
      P(M)=(Y(M)-A(M,  1)*P(M-1))/D
      IF (M.EQ.N)  GO TO 1
      H(M)=-A(M,  3)/D
    1 CONTINUE
      X(N)=P(N)
      DO 2  I=2,N
      M=N+1-I
    2 X(M)=P(M)+H(M)*X(M+1)
      RETURN
C        IN CASE OF ANY IMPENDING ZERO DIVISORS:
   10 WRITE (6,11)
   11 FORMAT ('0FIRST ELEMENT OF A, A(1,1), IS ZERO.',/,
     X   5X,'BETTER FIX IT BOSS.')
      GO TO 30
   20 WRITE (6,21) M
   21 FORMAT ('0ZERO DIVISOR ENCOUNTERED IN EQUATION M =',I3,'.',/,
     X   5X,'BETTER FIX IT BOSS.')
   30 WRITE (6,31)
   31 FORMAT ('0EXECUTION TERMINATED.')
      CALL EXIT
      RETURN
      END
```

```
      SUBROUTINE NORMAL(PHI,N)
C     NORMALIZES THE GROUP FLUXES TO ONE. NOT BOTH GROUPS.
      REAL*8 PHI(2,201), A
      A=DABS(PHI(1,1))
      DO 1  IG=1,2
      DO 1  I=1,N
      IF (DABS(PHI(IG,I)).GT.A)  A=DABS(PHI(IG,I))
    1 CONTINUE
      DO 2  IG=1,2
      DO 2  I=1,N
    2 PHI(IG,I)=PHI(IG,I)/A
      RETURN
      END
```

```
      SUBROUTINE NORM2(PSI,N)                                           NOR20001
C     NORMALIZES BOTH ENERGY GROUPS OF PSI TO 1.0.                      NOR20002
      REAL*8 PSI(2,201), A(2)                                           NOR20003
      DO 1  IG=1,2                                                      NOR20004
      A(IG)=DABS(PSI( IG,1))                                            NOR20005
      DO 1  I=1,N                                                       NOR20006
      IF (DABS(PSI(IG,I)).GT.A(IG))  A(IG)=DABS(PSI(IG,I))              NOR20007
    1 CONTINUE                                                          NOR20008
      DO 2  IG=1,2                                                      NOR20009
      DO 2  I=1,N                                                       NOR20010
    2 PSI(IG,I)=PSI(IG,I)/A(IG)                                         NOR20011
      RETURN                                                            NOR20012
      END                                                               NOR20013
```

```
      SUBROUTINE PHIPLT(L)                                              PHIP0001
C        PLOTS THE GROUP FLUX HISTORY, WITH UP TO 5 GROUP FLUXES PER PLOT.PHIP0002
C        FAST AND THERMAL GROUP FLUXES ARE PLOTTED SEPERATELY.          PHIP0003
C        L IS THE NUMBER OF FLUXES TO BE PLOTTED.                       PHIP0004
C        L IS BETWEEN 1 AND 5.                                          PHIP0005
      IMPLICIT REAL*8 (A-H,O-Z)                                         PHIP0006
      COMMON /B1/ IBC                                                   PHIP0007
      COMMON /B2/ KR,N                                                  PHIP0008
      COMMON /B5/ S(201), A(201,6), B(201,6)                            PHIP0009
      COMMON /B6/ TE1(2,5),TE2(2,5),TE3(5),IN(5)                        PHIP0010
      COMMON /ER/ EPS1,EPS2,EPS3                                        PHIP0011
      COMMON /FSTR/ PHISTR(2,201,6)                                     PHIP0012
      DIMENSION SYMBOL(5)                                               PHIP0013
      INTEGER SYMBOL /'.','-','+','#','*'/                              PHIP0014
      ND=201                                                            PHIP0015
      KR1=KR+1                                                          PHIP0016
C        SET UP B.C. CONDITIONS                                         PHIP0017
      IF (IBC.EQ.4)  GO TO 5                                            PHIP0018
      IF (IBC.EQ.3)  GO TO 3                                            PHIP0019
      DO 2  IG=1,2                                                      PHIP0020
      DO 2  K=1,L                                                       PHIP0021
      DO 1  I=1,N                                                       PHIP0022
      J=N+1-I                                                           PHIP0023
    1 PHISTR(IG,J+1,K+1)=PHISTR(IG,J,K+1)                               PHIP0024
    2 PHISTR(IG,1,K+1)=0.                                               PHIP0025
    3 IF (IBC.EQ.2)  GO TO 5                                            PHIP0026
      DO 4  IG=1,2                                                      PHIP0027
      DO 4  K=1,L                                                       PHIP0028
    4 PHISTR(IG,KR1,K+1)=0.                                             PHIP0029
    5 CONTINUE                                                          PHIP0030
C        FLUXES IN PHISTR HAVE BEEN NORMALIZED IN POWER.                PHIP0031
C        PUT THE FAST FLUX IN A, AND THE THERMAL FLUX IN B:             PHIP0032
      L1=L+1                                                            PHIP0033
      DO 10  K=1,L1                                                     PHIP0034
      DO 10  I=1,KR1                                                    PHIP0035
      A(I,K)=PHISTR(1,I,K)                                              PHIP0036
```

```
   10 B(I,K)=PHISTR(2,I,K)                                                PHIP0037
C         PLOT THE L FAST FLUX SHAPES ON ONE GRAPH:                       PHIP0038
      CALL PRTPLT(0,A,KR1,L1,KR1,0,ND,6,2)                                PHIP0039
      WRITE (6,20)                                                        PHIP0040
   20 FORMAT (/,'0FAST FLUX ITERATION HISTORY PLOT.',/)                   PHIP0041
      WRITE (6,30)                                                        PHIP0042
   30 FORMAT (                                                            PHIP0043
     X   '0KEY:',5X,'SYMBOL',5X,'ITERATION NUMBER:',7X,'ERROR CRITERIA',  PHIP0044
     X   11X,'ERROR',13X,'TOLERANCE')                                     PHIP0045
      DO 35  I=1,L                                                        PHIP0046
   35 WRITE (6,40) SYMBOL(I),IN(I),TE1(1,I),EPS1,TE2(1,I),EPS2,           PHIP0047
     X             TE3(I),EPS3                                            PHIP0048
   40 FORMAT (/,12X,A1,15X,I3,16X,'FLUX',14X,1PD15.5,5X,1PD15.5,/,        PHIP0049
     X   47X,'MEAN SQ. FLUX',5X,1PD15.5,5X,1PD15.5,/,                     PHIP0050
     X   47X,'EIGENVALUE',8X,1PD15.5,5X,1PD15.5)                          PHIP0051
C                                                                         PHIP0052
C         PLOT THE L THERMAL FLUX SHAPES ON THE OTHER GRAPH:              PHIP0053
      CALL PRTPLT(0,B,KR1,L1,KR1,0,ND,6,2)                                PHIP0054
      WRITE (6,50)                                                        PHIP0055
   50 FORMAT (/,'0THERMAL FLUX ITERATION PLOT.',/)                        PHIP0056
      WRITE (6,30)                                                        PHIP0057
      DO 55  I=1,L                                                        PHIP0058
   55 WRITE (6,40) SYMBOL(I),IN(I),TE1(2,I),EPS1,TE2(2,I),EPS2,           PHIP0059
     X             TE3(I),EPS3                                            PHIP0060
      RETURN                                                              PHIP0061
      END                                                                 PHIP0062
```

```
      SUBROUTINE PRTPLT(NO,B,N,M,NL,NS,KX,JX,ISP)
C*****MODIFIED VERSION FROM THAT OF SSP OR ANY OTHER SOURCE *****
C        CONVERTS DOUBLE PRECISION B ARRAY TO REAL*4.
C           PLOT SEVERAL CROSS-VARIABLES VERSUS A BASE VARIABLE
C           NO - CHART NUMBER (3 DIGITS MAXIMUM)
C           B  - MATRIX OF DATA TO BE PLOTTED. FIRST COLUMN REPRESENTS
C                BASE VARIABLE AND SUCCESSIVE COLUMNS ARE THE CROSS-
C                VARIABLES (MAXIMUM IS 9).
C           N  - NUMBER OF ROWS IN MATRIX B
C           M  - NUMBER OF COLUMNS IN MATRIX B (EQUAL TO THE TOTAL
C                NUMBER OF VARIABLES). MAXIMUM IS 10.
C           NL - NUMBER OF LINES IN THE PLOT. IF 0 IS SPECIFIED, 50
C                LINES ARE USED. THE NUMBER OF LINES MUST BE EQUAL TO
C                OR GREATER THAN N
C                (USUALLY USE NL=N, AND ISP FOR SPACING.)
C           NS - CODE FOR SORTING THE BASE VARIABLE DATA IN ASCENDING
C                ORDER
C                  0   SORTING IS NOT NECESSARY (ALREADY IN ASCENDING
C                      ORDER).
C                  1   SORTING IS NECESSARY.
C            KX- DIMENSION OF B MATRIX FROM DIMENSION STATEMENT.
C                IT MUST BE OF THE FORM B(KX,JX)
C            JX- DIMENSION OF B MATRIX FROM DIMENSION STATEMENT.
C                IT MUST BE OF THE FORM B(KX,JX)
C           ISP- CODE FOR SPACING LINES WHILE PLOTTING:
C                  1   SINGLE SPACE
C                  2   DOUBLE SPACE
C                  3   TRIPLE SPACE
C                  ...ETC.
      REAL*8 B
      DIMENSION OUT(101),YPR(11),IANG(9),A( 1500),B(KX,JX)
      INTEGER IDUM/'1'/,IANG/'.','-','+','#','*','A','B','C','D'/
      INTEGER OUT
      I=1
      DO 39 J=1,M
      DO 39 K=1,N
```

```
      A(I)=B(K,J)
      I=I+1
39    CONTINUE
    1 FORMAT(1H1,60X,7H CHART ,I3,//)
    2 FORMAT(1H ,F11.4,5X,101A1)
    3 FORMAT(1H )
    7 FORMAT(1H ,16X,101H+         +         +         +         +
     1    +         +         +         +         +         +)
    8 FORMAT(1H ,9X,11F10.4)
      NLL=NL
      IF(NS) 16, 16, 10
C        SORT BASE VARIABLE DATA IN ASCENDING ORDER
   10 DO 15 I=1,N
      DO 14 J=I,N
      IF(A(I)-A(J)) 14, 14, 11
   11 L=I-N
      LL=J-N
      DO 12 K=1,M
      L=L+N
      LL=LL+N
      F=A(L)
      A(L)=A(LL)
   12 A(LL)=F
   14 CONTINUE
   15 CONTINUE
C        TEST NLL
   16 IF(NLL) 20, 18, 20
   18 NLL=50
C        PRINT TITLE
   20 WRITE(6,1)NO
C        DEVELOP BLANK AND DIGITS FOR PRINTING
      BLANK=0
C        FIND SCALE FOR BASE VARIABLE
      XSCAL=(A(N)-A(1))/(FLOAT(NLL-1))
C        FIND SCALE FOR CROSS-VARIABLES
      YMIN=1.0E75
```

```
      YMAX=-1.0E75                                                      PRTP0073
      M1=N+1                                                            PRTP0074
      M2=M*N                                                            PRTP0075
      DO 40 J=M1,M2                                                     PRTP0076
      IF (A(J).GT.YMAX) YMAX=A(J)                                       PRTP0077
      IF (A(J).LT.YMIN) YMIN=A(J)                                       PRTP0078
   40 CONTINUE                                                          PRTP0079
      YSCAL=(YMAX-YMIN)/100.0                                           PRTP0080
C        CHECK TO SEE IF THE SPREAD IN Y IS TOO SMALL FOR PLOTTING:     PRTP0081
      IF (YSCAL.EQ.0.0)  GO TO 100                                      PRTP0082
C        OTHERWISE, A DIVIDE CHECK WILL OCCUR AFTER STATEMENT 56.       PRTP0083
C        FIND BASE VARIABLE PRINT POSITION                              PRTP0084
      XB=A(1)                                                           PRTP0085
      L=1                                                               PRTP0086
      MY=M-1                                                            PRTP0087
      IT=ISP-1                                                          PRTP0088
      DO 80 I=1,NLL                                                     PRTP0089
      F=I-1                                                             PRTP0090
      XPR=XB+F*XSCAL                                                    PRTP0091
      IF(A(L)-XPR-XSCAL*0.5) 50,50,70                                   PRTP0092
C        FIND CROSS-VARIABLES                                           PRTP0093
   50 DO 55 IX=1,101                                                    PRTP0094
   55 OUT(IX)=BLANK                                                     PRTP0095
   57 CONTINUE                                                          PRTP0096
      DO 60 J=1,MY                                                      PRTP0097
   56 LL=L+J*N                                                          PRTP0098
      JP=((A(LL)-YMIN)/YSCAL)+1.0                                       PRTP0099
      OUT(JP)=IANG(J)                                                   PRTP0100
   60 CONTINUE                                                          PRTP0101
C        PRINT LINE AND CLEAR, OR SKIP                                  PRTP0102
      IF(L.EQ.N)  GO TO 61                                              PRTP0103
      L=L+1                                                             PRTP0104
      IF(A(L)-XPR-XSCAL*0.5)  57,57,61                                  PRTP0105
   61 CONTINUE                                                          PRTP0106
      WRITE(6,2)XPR,(OUT(IZ),IZ=1,101)                                  PRTP0107
      IF (IT.EQ.0)  GO TO 65                                            PRTP0108
```

```
      DO 64  IV=1,IT                                                    PRTP0109
   64 WRITE (6,3)                                                       PRTP0110
   65 GO TO 80                                                          PRTP0111
   70 WRITE(6,3)                                                        PRTP0112
   80 CONTINUE                                                          PRTP0113
C         PRINT CROSS-VARIABLES NUMBERS                                 PRTP0114
      WRITE(6,7)                                                        PRTP0115
      YPR(1)=YMIN                                                       PRTP0116
      DO 90 KN=1,9                                                      PRTP0117
   90 YPR(KN+1)=YPR(KN)+YSCAL*10.0                                      PRTP0118
      YPR(11)=YMAX                                                      PRTP0119
      WRITE(6,8)(YPR(IP),IP=1,11)                                       PRTP0120
      RETURN                                                            PRTP0121
  100 WRITE (6,101)                                                     PRTP0122
  101 FORMAT ('0NO PLOT IS GENERATED BECAUSE THE SPREAD IN THE Y VARIBLE PRTP0123
     X IS TOO SMALL.',/,10X,'(I.E. - EQUALS ZERO UNDER REAL*4.)',//,    PRTP0124
     X   5X, 'EXECUTION CONTINUING.')                                   PRTP0125
      RETURN                                                            PRTP0126
      END                                                               PRTP0127
```

```
      SUBROUTINE PUNCH                                                  PNCH0001
C        PUNCHES THE FLUX AND CURRENT AND ADJOINTS OUT AFTER CONVERGENCE.   PNCH0002
C        CALLED BY IPUNCH=1.                                            PNCH0003
      IMPLICIT REAL*8 (A-H,O-Z)                                         PNCH0004
      COMMON /B2/ KR                                                    PNCH0005
      COMMON /B4/ F(2,201), C(2,201)                                    PNCH0006
      COMMON /B47/ F7(2,201), C7(2,201)                                 PNCH0007
      WRITE (7,1) KR                                                    PNCH0008
    1 FORMAT (I5)                                                       PNCH0009
      N=KR+1                                                            PNCH0010
C        PUNCH OUT THE FAST FLUX:                                       PNCH0011
      WRITE (7,10) (F(1,J),C(1,J),F7(1,J),C7(1,J),J=1,N)                PNCH0012
C        PUNCH OUT THE THERMAL FLUX:                                    PNCH0013
      WRITE (7,10) (F(2,J),C(2,J),F7(2,J),C7(2,J),J=1,N)                PNCH0014
   10 FORMAT (4E20.7)                                                   PNCH0015
      RETURN                                                            PNCH0016
      END                                                               PNCH0017
```

F.2. SOURCE LISTING of Program LINEAR

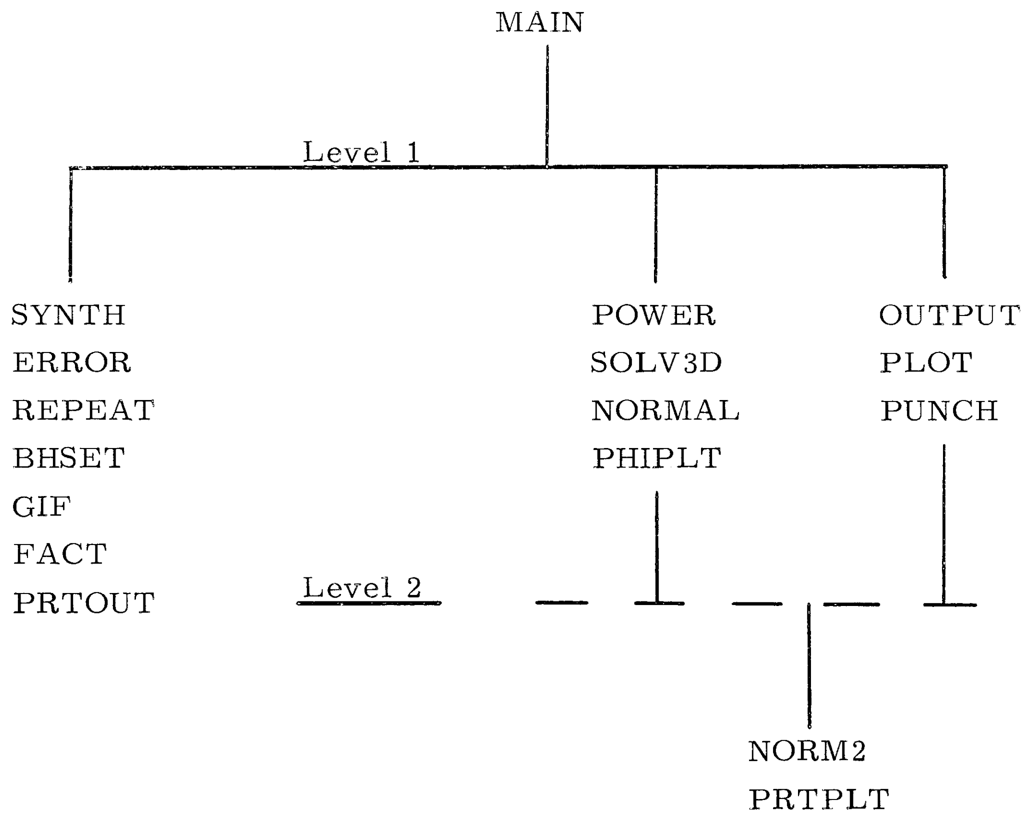


Figure F.2. Structure of Program LINEAR.

```
C     PROGRAM LINEAR:
C     TWO GROUP PROPOSED METHOD USING LINEAR BASIS FUNCTIONS.
      CALL TIMING(I1)
      CALL SYNTH
      CALL TIMING(I4)
      CALL POWER
      CALL TIMING(I6)
      CALL TIMING(I7)
      CALL OUTPUT
      CALL TIMING(I8)
C        TIMING EXECUTION
      WRITE (6,30)
   30 FORMAT (1H1,'TIMING PROGRAM EXECUTION:',/)
      J=I4-I1
      WRITE(6,701) J
      J=I6-I4
      WRITE(6,704) J
      J=I7-I6
      WRITE(6,706) J
      J=I8-I7
      WRITE(6,707) J
  701 FORMAT (1H ,' SYNTH HAS TAKEN',I6,' /100 SECONDS.')
  704 FORMAT (1H ,' POWER HAS TAKEN',I6,' /100 SECONDS.')
  706 FORMAT (1H ,' CURENT HAS TAKEN',I5,' /100 SECONDS.')
  707 FORMAT (1H ,' OUTPUT HAS TAKEN',I5,' /100 SECONDS.')
      CALL TIMING(I20)
      J=I20-I1
      WRITE(6,720) J
  720 FORMAT (1H0,' THIS RUN HAS TAKEN',I6,' /100 SECONDS TO RUN.')
      STOP
      END
```

```
      SUBROUTINE SYNTH
C        PROPOSED LINEAR SYNTHESIS METHOD:
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C        ADJOINT QUANTITIES OF VARIBLES ARE DENOTED BY 7 RATHER THAN *.
C        THUS:  PHI7 (RATHER THAN PHI*) IS THE ADJOINT OF PHI. ETC.
      IMPLICIT REAL*8 (A-H,K-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH,ISEE
      COMMON /B2/ KR,NN
      COMMON /B3/ L1(26,26),L2(26,26),F1(26,26),F2(26,26),
     X            F3(26,26),F4(26,26),T(26,26)
      COMMON /B5/  KA0(2,25),KA1(2,25),KA2(2,25),KB0(2,25),KB1(2,25),
     X             KB2(2,25),LA0(2,25),LA1(2,25),LA2(2,25),SR0(1,25),
     X             SR1(1,25),SR2(1,25),KC0(1,25),KC1(1,25),KC2(1,25),
     X             KD0(1,25),KD1(1,25),KD2(1,25),
     X             P(2,25),P1(2,25),Q(2,25),
     X             Q1(2,25),R(2,25),P0(2,25),P07(2,25),PH(2,25),
     X             PH7(2,25),AL(2,25),BL(2,25),CL(2,25),AF(4,25),
     X             BF(4,25),CF(4,25),AT(25),BT(25),CT(25),
     X             ALK(2),  BLK(2),  AFK(4),  BFK(4),  ATK(2),
     X             BTK(2),BL0(2),CL0(2),BF0(4),CF0(4),BT0(2),CT0(2),
     X             C0(2), CH(2)
      COMMON /CHIF/ CHI(2)
      COMMON /XAXIS/ HX, HR(25)
      COMMON /BH/ X(101),  H(101)
      COMMON /ER/ EPS1,EPS2,EPS3
      DIMENSION PHI(2,101),PHI7(2,101),CUR(2,101),CUR7(2,101),
     X          A(2,100),F(2,100),D(2,100),S(2,100),DI(2,100),
     X          XU(2,100)
      DIMENSION V(2),V1(2),V2(2),V3(2)
      DIMENSION ITF(25), KTF(25)
C        IN ORDER TO SAVE SPACE:
      EQUIVALENCE (PHI(1),L1(1)),      (PHI7(1),L1(301)),
     X            (CUR(1),L2(1)),      (CUR7(1), L2(301)),
     X            (XU(1),F1(1)),       (A(1),F1(301)),
     X            (F(1),F2(1)),        (D(1),F2(301)),
     X            (S(1),T(1)),         (DI(1),T(301))
```

```
      REAL TITLE(20)
      INTEGER KR,K,KS,KS1,KRO,NN,NUMITF,KTF
      READ (5,200) TITLE
  200 FORMAT (20A4)
      WRITE (6,201) TITLE
  201 FORMAT (1H1,20A4,//)
C        READ IN THE NUMBER OF REGION TRIAL FUNCTIONS AND TYPE OF B.C.S.
C        AS WELL AS THE TOLERANCES AND THE OUTPUT TYPES DESIRED:
      READ (5,1) KR,IBC,EPS1,EPS2,EPS3,IPLOT,JPLOT,IPUNCH,ISEE,ITW,ITC
    1 FORMAT (2I5,3D10.3,6I5)
      IF (IBC.EQ.3)  IBC=2
C        READ IN THE TYPE-NUMBER OF EACH TF REGION:
      READ (5,100) (ITF(I),I=1,KR)
  100 FORMAT (25I2)
C        READ IN THE FISSION YEILDS FOR EACH GROUP:
      READ (5,101) CHI(1), CHI(2)
  101 FORMAT (2F10.5)
      KRO=KR-1
      WRITE (6,2) KR, IBC, ISEE, ITW, ITC
    2 FORMAT ('OONE DIMENSIONAL TWO GROUP LINEAR SYNTHESIS PROGRAM:',//,
     X   5X,'NUMBER OF COARSE MESH REGIONS:  KR = ',I2,/,
     X   5X,'BOUNDARY CONDITION NUMBER:     IBC = ',I2,/,
     X   5X,'AMOUNT OF OUTPUT REQUESTED:   ISEE = ',I2,//,
     X   5X,'TYPE OF WEIGHTING FUNCTIONS:   ITW = ',I2,/,
     X   5X,'TYPE OF CURRENT FUNCTIONS:     ITC = ',I2,//,
     X   5X,'REGIONAL INPUT MATERIAL PROPERTIES AND FLUX SHAPES FOLLOW',
     X   /,5X,'IF ISEE > 0:',//,
     X   5X,'FLUX SHAPES ARE LINEAR IN EACH INDICATED SUBREGION.')
      IF (ITC.EQ.0)  WRITE (6,16)
      IF (ITC.EQ.1)  WRITE (6,17)
   16 FORMAT (5X,'CURRENTS ARE CONSTANT IN EACH INDICATED SUBREGION.')
   17 FORMAT (5X,'CURRENTS ARE LINEAR IN EACH INDICATED SUBREGION.')
      IF (ITW.EQ.0)  WRITE (6,116)
      IF (ITW.EQ.1)  WRITE (6,117)
  116 FORMAT (/,5X,'WEIGHTING FLUX = FLUX;',/,5X,'WEIGHTING CURRENT = -
     XCURRENT.')
```

```
  117 FORMAT (/,5X,'WEIGHTING FLUX = ADJOINT FLUX;',/,5X,'WEIGHTING CURR
     XENT = ADJOINT CURRENT.')
      WRITE (6,20) EPS1,EPS2,EPS3,IPLOT,JPLOT,IPUNCH
   20 FORMAT (//,'0TOLERANCES TO POWER ARE :  EPS1 = ',1PD10.3,/,
     X    28X,'EPS2 = ',1PD10.3,/,28X,'EPS3 = ',1PD10.3,/,
     X    '0OUTPUT PARAMETERS TO POWER ARE:  IPLOT = ',I1,/,
     X    34X,'JPLOT = ',I1,/,34X,'IPUNCH = ',I1)
      WRITE (6,22) CHI(1), CHI(2)
   22 FORMAT (/,'0FISSION YIELDS ARE:  CHI(1) =',F10.5,/,
     X    22X,'CHI(2) =',F10.5)
      IF ((KR.LE.2).AND.(IBC.EQ.1))  CALL ERROR(1,KR)
      IF (KR.GT.25)  CALL ERROR(2,KR)
      IF (EPS1.LT.1.0E-16)  CALL ERROR(6,1)
      IF (EPS2.LT.1.0E-16)  CALL ERROR(6,2)
      IF (EPS3.LT.1.0E-16)  CALL ERROR(6,3)
      IF ((IBC.LT.1).OR.(IBC.GT.7))  CALL ERROR(7,IBC)
C        DUMMY NORMAL VECTOR XU = UNITY.  (FOR THE INTEGRATION FUNCTIONS)
      DO 21  IG=1,2
      DO 21  II=1,100
   21 XU(IG,II)=1.0
      ITC0=2
      ITC1=2
      IF (ITC.EQ.1)  GO TO 23
      ITC0=0
      ITC1=1
C        COUNTER OF THE NUMBER OF TYPE-NUMBERS OF EACH TF REGION:
   23 NUMITF=1
      HX=0.0
C        BEGIN TO READ IN THE TF REGION DATA AND FILL THE ARRAYS,
C        DEPENDING ON THE TYPE-NUMBER OF EACH TF REGION.
      DO 50  I=1,KR
      IF (ITF(I).EQ.NUMITF)  GO TO 110
C        FILL THE ARRAYS FROM OLD TF REGION TYPES:
      J=ITF(I)
      CALL REPEAT(I,KTF(J))
      GO TO 50
```

```
C         READ IN THE TF REGION'S DATA FOR NEW TF REGION TYPE-NUMBERS:          SYNT0109
  110 NUMITF=NUMITF+1                                                           SYNT0110
      KTF(NUMITF-1)=I                                                           SYNT0111
C         READ THE SUBREGION NUMBER AND THE NUMBER OF REGIONS IN THE SUBREGION:  SYNT0112
      READ (5,1) K, KS                                                          SYNT0113
      IF (KS.GT.100)  CALL ERROR(3,I)                                           SYNT0114
C         CHECK FOR IMPROPER SEQUENCING OF INPUT DATA:                          SYNT0115
      IF (I.NE.K)  CALL ERROR(4,I)                                              SYNT0116
C         READ IN THE GEOMETRY AND THE MATERIAL PROPERTIES:                     SYNT0117
      READ (5,3) (X(J),X(J+1),H(J),A(1,J),F(1,J),D(1,J),S(1,J),                 SYNT0118
     X                            A(2,J),F(2,J),D(2,J),J=1,KS)                  SYNT0119
    3 FORMAT (3F10.5,4D10.3,/,30X,3D10.3)                                       SYNT0120
C         READ IN THE REGIONAL GROUP TRIAL FUNCTIONS:                           SYNT0121
      KS1=KS+1                                                                  SYNT0122
      READ (5,4) (PHI(1,J),CUR(1,J),PHI7(1,J),CUR7(1,J),J=1,KS1)               SYNT0123
      READ (5,4) (PHI(2,J),CUR(2,J),PHI7(2,J),CUR7(2,J),J=1,KS1)               SYNT0124
    4 FORMAT (4D20.7)                                                           SYNT0125
      IF (ITW.EQ.1)  GO TO 120                                                  SYNT0126
C         FORM WEIGHTING FUNCTIONS FROM THE GIVEN FUNCTIONS:                    SYNT0127
      DO 119  IG=1,2                                                            SYNT0128
      DO 119  J=1,KS1                                                           SYNT0129
      PHI7(IG,J)=PHI(IG,J)                                                      SYNT0130
  119 CUR7(IG,J)=-CUR(IG,J)                                                     SYNT0131
  120 IF (ITC.EQ.1)  GO TO 5                                                    SYNT0132
C         FORM THE REGION CONSTANT CURRENTS:                                    SYNT0133
      DO 7  IG=1,2                                                              SYNT0134
      DO 6  J=1,KS                                                              SYNT0135
      CUR(IG,J)=-D(IG,J)*(-PHI(IG,J)+PHI(IG,J+1))/H(J)                          SYNT0136
    6 CUR7(IG,J)=+D(IG,J)*(-PHI7(IG,J)+PHI7(IG,J+1))/H(J)                       SYNT0137
      CUR(IG,KS1)=0.0                                                           SYNT0138
    7 CUR7(IG,KS1)=0.0                                                          SYNT0139
C         WRITING OUT THE INPUT INFORMATION:                                    SYNT0140
    5 IF (ISEE.EQ.0)  GO TO 14                                                  SYNT0141
      WRITE (6,10)  K,KR,KS,(J,X(J),X(J+1),H(J),A(1,J),F(1,J),D(1,J),           SYNT0142
     X   S(1,J),A(2,J),F(2,J),D(2,J),J=1,KS)                                    SYNT0143
   10 FORMAT ('1INPUT MATERIAL PROPERTIES FOR REGION NUMBER ',I3,               SYNT0144
```

```
     X   ', OF THE ',I3,' USED.',//,
     X   5X,'THIS REGION IS DIVIDED INTO ',I3,' HOMOGENEOUS SUBREGIONS A
     XS FOLLOWS:',//,
     X   5X,'FAST GROUP CONSTANTS APPEAR FIRST:',//,
     X   ' SUBREGION #',5X,'INTERNAL BOUNDARIES',10X,'WIDTH',3X,
     X   '  TOTAL CX (1/CM)',3X,'FISSION CX (1/CM)',6X,'DIFFUSION (CM)',
     X   4X,'SCATT.CX (1/CM)',/,
     X   5X,'I',11X,'X(I)',9X,'X(I+1)',11X,'H(I)',13X,'A(IG,I)',13X,
     X   'F(IG,I)',13X,'D(IG,I)',14X,'S(1,I)',//,
     X   (I6,3F15.4,4D20.8,/,51X,3D20.8))
C
      DO 15  IG=1,2
   15 WRITE (6,11) IG,K,KR,(J,X(J),PHI(IG,J),CUR(IG,J),PHI7(IG,J),
     X   CUR7(IG,J),J=1,KS1)
   11 FORMAT ('1INPUT TRIAL FUNCTIONS FOR GROUP',I2,' FOR REGION',I3,
     X   ' OUT OF THE',I3,' USED:',//,
     X   ' INDEX',5X,'COORD',16X,'FLUX',13X,'CURRENT',8X,' WEIGHT FLUX',
     X   5X,' WEIGHT CURRENT',//,(I6,F10.5,4D20.7))
   14 CONTINUE
C        END OF THE IN-OUT SECTION:
C        DEFINING MISC. ARRAYS FOR THE INTEGRATION FUNCTIONS:
C        LEGNTH OF THE SUBREGION: HT
      HT=X(KS1)-X(1)
      HR(K)=HT
      HX=HX+HR(K)
C        INVERSE OF THE D ARRAYS:
      DO 13  J=1,KS
      DI(1,J)=1./D(1,J)
   13 DI(2,J)=1./D(2,J)
C        FORMATION OF THE INTEGRATION FUNCTIONS:
      CALL BHSET(KS)
C        DO FOR ALL ENERGY GROUPS:
      DO 50  IG=1,2
      KA0(IG,K)=GIF(0,IG,PHI7,IG,A,PHI,KS,2)
      KA1(IG,K)=GIF(1,IG,PHI7,IG,A,PHI,KS,2)
      KA2(IG,K)=GIF(2,IG,PHI7,IG,A,PHI,KS,2)
```

```
      KB0(IG,K)=GIF(0,IG,PHI7,IG,F,PHI,KS,2)
      KB1(IG,K)=GIF(1,IG,PHI7,IG,F,PHI,KS,2)
      KB2(IG,K)=GIF(2,IG,PHI7,IG,F,PHI,KS,2)
      LA0(IG,K)=GIF(0,IG,CUR7,IG,DI,CUR,KS,ITC0)
      LA1(IG,K)=GIF(1,IG,CUR7,IG,DI,CUR,KS,ITC0)
      LA2(IG,K)=GIF(2,IG,CUR7,IG,DI,CUR,KS,ITC0)
      P(IG,K)  =GIF(0,IG,PHI7,IG,XU,CUR,KS,ITC1)/HT
      P1(IG,K) =GIF(1,IG,PHI7,IG,XU,CUR,KS,ITC1)/HT
      Q(IG,K)  =GIF(0,IG,PHI,IG,XU,CUR7,KS,ITC1)/HT
      Q1(IG,K) =GIF(1,IG,PHI,IG,XU,CUR7,KS,ITC1)/HT
      R(IG,K)  =GIF(0,IG,PHI7,IG,D,PHI,KS,2)/HT**2
C        STORE THE TERMINAL POINTS FOR LATER USE:
      P0(IG,K)=PHI(IG,1)
      P07(IG,K)=PHI7(IG,1)
      PH(IG,K)=PHI(IG,KS1)
      PH7(IG,K)=PHI7(IG,KS1)
      IF (K.EQ.1)  C0(IG)=CUR(IG,1)
      IF (NUMITF-1.EQ.ITF(KR).AND.ITC.EQ.0)  CH(IG)=CUR(IG,KS)
      IF (NUMITF-1.EQ.ITF(KR).AND.ITC.EQ.1)  CH(IG)=CUR(IG,KS1)
C        FOR THE OFF DIAGONAL MATRIX ELEMENTS:
      IF (IG.EQ.2)  GO TO 50
      SR0(IG,K)=GIF(0,2,PHI7,1,S,PHI,KS,2)
      SR1(IG,K)=GIF(1,2,PHI7,1,S,PHI,KS,2)
      SR2(IG,K)=GIF(2,2,PHI7,1,S,PHI,KS,2)
      KC0(IG,K)=GIF(0,1,PHI7,2,F,PHI,KS,2)
      KC1(IG,K)=GIF(1,1,PHI7,2,F,PHI,KS,2)
      KC2(IG,K)=GIF(2,1,PHI7,2,F,PHI,KS,2)
      KD0(IG,K)=GIF(0,2,PHI7,1,F,PHI,KS,2)
      KD1(IG,K)=GIF(1,2,PHI7,1,F,PHI,KS,2)
      KD2(IG,K)=GIF(2,2,PHI7,1,F,PHI,KS,2)
   50 CONTINUE
      NUMITF=NUMITF-1
      WRITE (6,51) NUMITF
   51 FORMAT ('1THERE ARE ONLY',I3,' DIFFERENT TRIAL FUNCTION REGIONS.')
      WRITE (6,52) (I,ITF(I),I=1,KR)
   52 FORMAT (/,'0TABLE OF THE TRIAL FUNCTION NUMBER TYPES:',//,
```

```
     X   3X,'TF REGION',4X,'REGION TYPE-NUMBER',//,                    SYNT0217
     X   (I7,12X,I7))                                                  SYNT0218
C        PRINTS OUT THE /B5/ ARRAYS:                                   SYNT0219
      IF (ISEE.GE.2)  CALL PRTOUT(1)                                   SYNT0220
C        NN IS THE MATRIX BLOCK SIZE:                                  SYNT0221
      NN=KR                                                            SYNT0222
      IF (IBC.EQ.1.OR.IBC.GE.6)  NN=KR-1                               SYNT0223
      IF (IBC.EQ.4)  NN=KR+1                                           SYNT0224
C        FORMATION OF THE COEFFICIENT VECTORS:                         SYNT0225
C        THE INTERIOR COEFFICIENTS:                                    SYNT0226
      IX=2                                                             SYNT0227
      IY=KR                                                            SYNT0228
      IF (IBC.EQ.5.OR.IBC.EQ.7)  IX=3                                  SYNT0229
      IF (IBC.EQ.1.OR.IBC.GE.6)  IY=KR-1                               SYNT0230
      DO 60  IG=1,2                                                    SYNT0231
      DO 60  K=IX,IY                                                   SYNT0232
      J=K-1                                                            SYNT0233
      V(IG)=1./(PH7(IG,J)*PO(IG,J))                                    SYNT0234
      V1(IG)=1./(PH7(IG,J)*PH(IG,J))                                   SYNT0235
      V2(IG)=1./(PO7(IG,K)*PO(IG,K))                                   SYNT0236
      V3(IG)=1./(PO7(IG,K)*PH(IG,K))                                   SYNT0237
      AL(IG,K)=(KA1(IG,J)-KA2(IG,J)-R(IG,J)+LA2(IG,J)-LA1(IG,J)        SYNT0238
     X   -Q1(IG,J)+P1(IG,J)-P(IG,J))*V(IG)                             SYNT0239
      BL(IG,K)=(KA2(IG,J)-LA2(IG,J)+Q1(IG,J)-P1(IG,J)+R(IG,J))*V1(IG)  SYNT0240
     X   +(KA0(IG,K)-2.*KA1(IG,K)+KA2(IG,K)-LA0(IG,K)+2.*LA1(IG,K)     SYNT0241
     X   -LA2(IG,K)+Q1(IG,K)-Q(IG,K)+P(IG,K)-P1(IG,K)+R(IG,K))*V2(IG)  SYNT0242
      CL(IG,K)=(KA1(IG,K)-KA2(IG,K)-R(IG,K)+LA2(IG,K)-LA1(IG,K)+Q(IG,K) SYNT0243
     X   -Q1(IG,K)+P1(IG,K))*V3(IG)                                    SYNT0244
      AF(IG,K)=(KB1(IG,J)-KB2(IG,J))*V(IG)                             SYNT0245
      BF(IG,K)=KB2(IG,J)*V1(IG)+(KB0(IG,K)-2.*KB1(IG,K)+KB2(IG,K))     SYNT0246
     X   *V2(IG)                                                       SYNT0247
      CF(IG,K)=(KB1(IG,K)-KB2(IG,K))*V3(IG)                            SYNT0248
      IF (IG.EQ.2)  GO TO 60                                           SYNT0249
      AT(K)=(SR1(1,J)-SR2(1,J))/(PH7(2,J)*PO(1,J))                     SYNT0250
      AF(3,K)=(KC1(IG,J)-KC2(IG,J))/(PH7(1,J)*PO(2,J))                 SYNT0251
      AF(4,K)=(KD1(IG,J)-KD2(IG,J))/(PH7(2,J)*PO(1,J))                 SYNT0252
```

```
      BT(K)=SR2(1,J)/(PH7(2,J)*PH(1,J))
     X    +(SR0(1,K)-2.*SR1(1,K)+SR2(1,K))/(P07(2,K)*P0(1,K))
      BF(3,K)=KC2(IG,J)/(PH7(1,J)*PH(2,J))
     X    +(KC0(IG,K)-2.*KC1(IG,K)+KC2(IG,K))/(P07(1,K)*P0(2,K))
      BF(4,K)=KD2(IG,J)/(PH7(2,J)*PH(1,J))
     X    +(KD0(IG,K)-2.*KD1(IG,K)+KD2(IG,K))/(P07(2,K)*P0(1,K))
      CT(K)=(SR1(1,K)-SR2(1,K))/(P07(2,K)*PH(1,K))
      CF(3,K)=(KC1(IG,K)-KC2(IG,K))/(P07(1,K)*PH(2,K))
      CF(4,K)=(KD1(IG,K)-KD2(IG,K))/(P07(2,K)*PH(1,K))
   60 CONTINUE
C         THE ZERO FLUX COEFFS:
C         NONE NEEDED AS F(1,1) AND F(2,1) BOTH = 0.0.
      IF (IBC.EQ.1)  GO TO 64
C
      IF (.NOT.(IBC.EQ.5.OR.IBC.EQ.7))  GO TO 66
C         ZERO FLUX COEFFICIENTS FOR NO TILTING ON THE LEFT:
      DO 61  IG=1,2
      V(IG)=1./(PH7(IG,1)*PH(IG,1))
      V1(IG)=1./(P07(IG,2)*P0(IG,2))
      V2(IG)=1./(P07(IG,2)*PH(IG,2))
      BL0(IG)=(KA0(IG,1)-LA0(IG,1))*V(IG)+(KA0(IG,2)-2.*KA1(IG,2)
     X    +KA2(IG,2)-LA0(IG,2)+2.*LA1(IG,2)-LA2(IG,2)+P(IG,2)-P1(IG,2)
     X    -Q(IG,2)+Q1(IG,2)+R(IG,2))*V1(IG)
      CL0(IG)=(KA1(IG,2)-KA2(IG,2)+LA2(IG,2)-LA1(IG,2)+P1(IG,2)
     X    +Q(IG,2)-Q1(IG,2)-R(IG,2))*V2(IG)
      BF0(IG)=KB0(IG,1)*V(IG)+(KB0(IG,2)-2.*KB1(IG,2)+KB2(IG,2))*V1(IG)
   61 CF0(IG)=(KB1(IG,2)-KB2(IG,2))*V2(IG)
      BT0(1)=SR0(1,1)/(PH7(2,1)*PH(1,1))
     X    +(SR0(1,2)-2.*SR1(1,2)+SR2(1,2))/(P07(2,2)*P0(1,2))
      BF0(3)=KC0(1,1)/(PH7(1,1)*PH(2,1))
     X    +(KC0(1,2)-2.*KC1(1,2)+KC2(1,2))/(P07(1,2)*P0(2,2))
      BF0(4)=KD0(1,1)/(PH7(2,1)*PH(1,1))
     X    +(KD0(1,2)-2.*KD1(1,2)+KD2(1,2))/(P07(2,2)*P0(1,2))
      CT0(1)=(SR1(1,2)-SR2(1,2))/(P07(2,2)*PH(1,2))
      CF0(3)=(KC1(1,2)-KC2(1,2))/(P07(1,2)*PH(2,2))
      CF0(4)=(KD1(1,2)-KD2(1,2))/(P07(2,2)*PH(1,2))
```

```
   66 IF (IBC.LE.5)  GO TO 69                                          SYNT0289
C         FOR THE LAST REGION TO BE NOT TILTED:                        SYNT0290
      K=KR-1                                                           SYNT0291
      DO 67  IG=1,2                                                    SYNT0292
      V(IG)=1./(PH7(IG,K)*PO(IG,K))                                    SYNT0293
      V1(IG)=1./(PO7(IG,KR)*PO(IG,KR))                                 SYNT0294
      V2(IG)=1./(PH7(IG,K)*PH(IG,K))                                   SYNT0295
      AL(IG,KR)=(KA1(IG,K)-KA2(IG,K)-LA1(IG,K)+LA2(IG,K)-P(IG,K)       SYNT0296
     X   +P1(IG,K)-Q1(IG,K)-R(IG,K))*V(IG)                             SYNT0297
      BL(IG,KR)=(KA0(IG,KR)-LA0(IG,KR))*V1(IG)+(KA2(IG,K)-LA2(IG,K)    SYNT0298
     X   -P1(IG,K)+Q1(IG,K)+R(IG,K))*V2(IG)                            SYNT0299
      AF(IG,KR)=(KB1(IG,K)-KB2(IG,K))*V(IG)                            SYNT0300
      BF(IG,KR)=KB0(IG,KR)*V1(IG)+KB2(IG,K)*V2(IG)                     SYNT0301
      IF (IG.EQ.2)  GO TO 67                                           SYNT0302
      AT(   KR)=(SR1(IG,K)-SR2(IG,K))/(PH7(2,K)*PO(1,K))               SYNT0303
      AF(3,KR)=(KC1(IG,K)-KC2(IG,K))/(PH7(1,K)*PO(2,K))                SYNT0304
      AF(4,KR)=(KD1(IG,K)-KD2(IG,K))/(PH7(2,K)*PO(1,K))                SYNT0305
      BT(KR)=SR0(IG,KR)/(PO7(2,KR)*PO(1,KR))                           SYNT0306
     X   +SR2(IG,K)/(PH7(2,K)*PH(1,K))                                 SYNT0307
      BF(3,KR)=KC0(IG,KR)/(PO7(1,KR)*PO(2,KR))                         SYNT0308
     X   +KC2(IG,K)/(PH7(1,K)*PH(2,K))                                 SYNT0309
      BF(4,KR)=KD0(IG,KR)/(PO7(2,KR)*PO(1,KR))                         SYNT0310
     X   +KD2(IG,K)/(PH7(2,K)*PH(1,K))                                 SYNT0311
   67 CONTINUE                                                         SYNT0312
      GO TO 64                                                         SYNT0313
C         THE ZERO CURRENT COEFFS:                                     SYNT0314
   69 K=KR                                                             SYNT0315
      DO 62  IG=1,2                                                    SYNT0316
      V(IG)=1./(PH7(IG,K)*PO(IG,K))                                    SYNT0317
      V1(IG)=1./(PH7(IG,K)*PH(IG,K))                                   SYNT0318
      ALK(IG)=(KA1(IG,K)-KA2(IG,K)-R(IG,K)+P1(IG,K)-P(IG,K)+LA2(IG,K)  SYNT0319
     X   -LA1(IG,K)-Q1(IG,K))*V(IG)                                    SYNT0320
      BLK(IG)=(KA2(IG,K)+R(IG,K)-P1(IG,K)-LA2(IG,K)+Q1(IG,K))*V1(IG)   SYNT0321
     X   +CH(IG)/PH(IG,KR)                                             SYNT0322
      AFK(IG)=(KB1(IG,K)-KB2(IG,K))*V(IG)                              SYNT0323
   62 BFK(IG)=KB2(IG,K)*V1(IG)                                         SYNT0324
```

```
      ATK(1)=(SR1(1,K)-SR2(1,K))/(PH7(2,K)*PO(1,K))                    SYNT0325
      AFK(3)=(KC1(1,K)-KC2(1,K))/(PH7(1,K)*PO(2,K))                    SYNT0326
      AFK(4)=(KD1(1,K)-KD2(1,K))/(PH7(2,K)*PO(1,K))                    SYNT0327
      BTK(1)=SR2(1,K)/(PH7(2,K)*PH(1,K))                               SYNT0328
      BFK(3)=KC2(1,K)/(PH7(1,K)*PH(2,K))                               SYNT0329
      BFK(4)=KD2(1,K)/(PH7(2,K)*PH(1,K))                               SYNT0330
      IF (IBC.NE.4)  GO TO 64                                          SYNT0331
C         ZERO CURRENT ON THE LEFT COEFFS:                             SYNT0332
      K=1                                                              SYNT0333
      DO 63  IG=1,2                                                    SYNT0334
      V2(IG)=1./(PO7(IG,K)*PO(IG,K))                                   SYNT0335
      V3(IG)=1./(PO7(IG,K)*PH(IG,K))                                   SYNT0336
      BLO(IG)=                                                         SYNT0337
     X    +(KAO(IG,K)-2.*KA1(IG,K)+KA2(IG,K)-LAO(IG,K)+2.*LA1(IG,K)    SYNT0338
     X    -LA2(IG,K)+Q1(IG,K)-Q(IG,K)+P(IG,K)-P1(IG,K)+R(IG,K))*V2(IG) SYNT0339
     X    -CO(IG)/PO(IG,1)                                             SYNT0340
      CLO(IG) =(KA1(IG,K)-KA2(IG,K)-R(IG,K)+LA2(IG,K)-LA1(IG,K)+Q(IG,K)SYNT0341
     X    -Q1(IG,K)+P1(IG,K))*V3(IG)                                   SYNT0342
      BFO(IG)=            V2(IG)*(KBO(IG,K)-2.*KB1(IG,K)+KB2(IG,K))    SYNT0343
   63 CFO(IG) =(KB1(IG,K)-KB2(IG,K))*V3(IG)                            SYNT0344
      BTO(1)=(SRO(1,K)-2.*SR1(1,K)+SR2(1,K))/(PO7(2,K)*PO(1,K))        SYNT0345
      BFO(3)=(KCO(1,K)-2.*KC1(1,K)+KC2(1,K))/(PO7(1,K)*PO(2,K))        SYNT0346
      BFO(4)=(KDO(1,K)-2.*KD1(1,K)+KD2(1,K))/(PO7(2,K)*PO(1,K))        SYNT0347
      CTO(1)=(SR1(1,K)-SR2(1,K))/(PO7(2,K)*PH(1,K))                    SYNT0348
      CFO(3)=(KC1(1,K)-KC2(1,K))/(PO7(1,K)*PH(2,K))                    SYNT0349
      CFO(4)=(KD1(1,K)-KD2(1,K))/(PO7(2,K)*PH(1,K))                    SYNT0350
C         ZERO MATRICES:                                               SYNT0351
   64 DO 65  I=1,NN                                                    SYNT0352
      DO 65  J=1,NN                                                    SYNT0353
      L1(I,J)=0.0                                                      SYNT0354
      L2(I,J)=0.0                                                      SYNT0355
      F1(I,J)=0.0                                                      SYNT0356
      F2(I,J)=0.0                                                      SYNT0357
      F3(I,J)=0.0                                                      SYNT0358
      F4(I,J)=0.0                                                      SYNT0359
   65  T(I,J)=0.0                                                      SYNT0360
```

```
C         FILLING THE MATRICES FOR POWER:                               SYNT0361
C         DETERMINE THE LEFT BC:                                        SYNT0362
      J=1                                                               SYNT0363
      IF (IBC.LE.2.OR.IBC.EQ.6)  GO TO 75                               SYNT0364
      L1(J,1)=BL0(1)                                                    SYNT0365
      L2(J,1)=BL0(2)                                                    SYNT0366
      F1(J,1)=BF0(1)                                                    SYNT0367
      F2(J,1)=BF0(2)                                                    SYNT0368
      F3(J,1)=BF0(3)                                                    SYNT0369
      F4(J,1)=BF0(4)                                                    SYNT0370
      T(J,1)=BT0(1)                                                     SYNT0371
      L1(J,2)=CL0(1)                                                    SYNT0372
      L2(J,2)=CL0(2)                                                    SYNT0373
      F1(J,2)=CF0(1)                                                    SYNT0374
      F2(J,2)=CF0(2)                                                    SYNT0375
      F3(J,2)=CF0(3)                                                    SYNT0376
      F4(J,2)=CF0(4)                                                    SYNT0377
      T(J,2)=CT0(1)                                                     SYNT0378
      J=J+1                                                             SYNT0379
C         FOR ALL INTERNAL EQUATIONS:                                   SYNT0380
   75 DO 70  K=IX,IY                                                    SYNT0381
      IF (J.EQ.1)  GO TO 76                                             SYNT0382
      L1(J,J-1)=AL(1,K)                                                 SYNT0383
      L2(J,J-1)=AL(2,K)                                                 SYNT0384
      F1(J,J-1)=AF(1,K)                                                 SYNT0385
      F2(J,J-1)=AF(2,K)                                                 SYNT0386
      F3(J,J-1)=AF(3,K)                                                 SYNT0387
      F4(J,J-1)=AF(4,K)                                                 SYNT0388
      T(J,J-1)=AT(K)                                                    SYNT0389
   76 L1(J,J)=BL(1,K)                                                   SYNT0390
      L1(J,J+1)=CL(1,K)                                                 SYNT0391
      L2(J,J)=BL(2,K)                                                   SYNT0392
      L2(J,J+1)=CL(2,K)                                                 SYNT0393
      F1(J,J)=BF(1,K)                                                   SYNT0394
      F1(J,J+1)=CF(1,K)                                                 SYNT0395
      F2(J,J)=BF(2,K)                                                   SYNT0396
```

```
      F2(J,J+1)=CF(2,K)                                                 SYNT0397
      F3(J,J)=BF(3,K)                                                   SYNT0398
      F3(J,J+1)=CF(3,K)                                                 SYNT0399
      F4(J,J)=BF(4,K)                                                   SYNT0400
      F4(J,J+1)=CF(4,K)                                                 SYNT0401
      T(J,J)=BT(K)                                                      SYNT0402
      T(J,J+1)=CT(K)                                                    SYNT0403
      J=J+1                                                             SYNT0404
   70 CONTINUE                                                          SYNT0405
      IF (IBC.EQ.1.OR.IBC.EQ.5)   GO TO 500                             SYNT0406
C         FOR THE LAST EQUATION:   (K=NN):                              SYNT0407
      K=NN                                                              SYNT0408
      L1(K ,K -1)=ALK(1)                                                SYNT0409
      L1(K ,K )=BLK(1)                                                  SYNT0410
      L2(K ,K -1)=ALK(2)                                                SYNT0411
      L2(K ,K )=BLK(2)                                                  SYNT0412
      F1(K ,K -1)=AFK(1)                                                SYNT0413
      F1(K ,K )=BFK(1)                                                  SYNT0414
      F2(K ,K -1)=AFK(2)                                                SYNT0415
      F2(K ,K )=BFK(2)                                                  SYNT0416
      F3(K,K-1)=AFK(3)                                                  SYNT0417
      F3(K,K)=BFK(3)                                                    SYNT0418
      F4(K,K-1)=AFK(4)                                                  SYNT0419
      F4(K,K)=BFK(4)                                                    SYNT0420
      T(K ,K -1)=ATK(1)                                                 SYNT0421
      T(K ,K )=BTK(1)                                                   SYNT0422
  500 CONTINUE                                                          SYNT0423
C         PRINTS OUT THE /B3/ MATRICES:                                 SYNT0424
      IF (ISEE.GE.2)  CALL PRTOUT(2)                                    SYNT0425
      RETURN                                                            SYNT0426
      END                                                               SYNT0427
```

```
      SUBROUTINE ERROR(I,J)
C        ANNOUNCES INPUT ERRORS AND TERMINATES PROGRAM EXECUTION:
      GO TO (1,2,3,4,5,6,7,8,9),I
    1 WRITE (6,101)
      GO TO 10
    2 WRITE (6,102) J
      GO TO 10
    3 WRITE (6,103) J
      GO TO 10
    4 WRITE (6,104) J
      GO TO 10
    5 WRITE (6,105) J
      GO TO 10
    6 WRITE (6,106) J
      GO TO 10
    7 CONTINUE
    8 CONTINUE
    9 CONTINUE
   10 WRITE (6,110)
  101 FORMAT ('1MUST HAVE > 2 SUBREGICNS FCR ZERO FLUX B.C.S. INVALID.')
  102 FORMAT ('1NUMBER OF SUBREGIONS =',I3,' > 25. INVALID.')
  103 FORMAT ('1SUBREGICN NUMBER',I3,' HAS > 25 SECTIONS. INVALID.')
  104 FORMAT ('1INPUT ERROR IN REGICN SEQUENCING AT REGION',I5,'.')
  105 FORMAT ('1Z(I) = 0. IN REGICN I =',I3,'. INVALID.')
  106 FORMAT ('1THE TOLERANCE:  EPS',I1,'  IS < 1.0E-16.  INVALID.')
  107 FORMAT ('1BOUNDRY CONDITION OPTICN =',I2,' < 1 OR > 7. INVALID.')
  110 FORMAT (1H0,'PROBLEM TERMINATED.')
      CALL EXIT
      RETURN
      END
```

```
      SUBROUTINE REPEAT(K,L)
C     SETS THE /B5/ ARRAYS (K) EQUAL TO PAST STORED ARRAYS (L):
      IMPLICIT REAL*8 (A-Z)
      COMMON /B5/  KA0(2,25),KA1(2,25),KA2(2,25),KB0(2,25),KB1(2,25),
     X             KB2(2,25),LA0(2,25),LA1(2,25),LA2(2,25),SR0(1,25),
     X             SR1(1,25),SR2(1,25),KC0(1,25),KC1(1,25),KC2(1,25),
     X             KD0(1,25),KD1(1,25),KD2(1,25),
     X             P(2,25),P1(2,25),Q(2,25),
     X             Q1(2,25), R(2,25),  P0(2,25), P07(2,25),PH(2,25),
     X             PH7(2,25)
      COMMON /XAXIS/ HX,HR(25)
      INTEGER K,L,G
      HR(K)=HR(L)
      HX=HX+HR(K)
      DO 10  G=1,2
      KA0(G,K)=KA0(G,L)
      KA1(G,K)=KA1(G,L)
      KA2(G,K)=KA2(G,L)
      KB0(G,K)=KB0(G,L)
      KB1(G,K)=KB1(G,L)
      KB2(G,K)=KB2(G,L)
      LA0(G,K)=LA0(G,L)
      LA1(G,K)=LA1(G,L)
      LA2(G,K)=LA2(G,L)
      IF (G.EQ.2)  GO TO 5
      SR0(G,K)=SR0(G,L)
      SR1(G,K)=SR1(G,L)
      SR2(G,K)=SR2(G,L)
      KC0(G,K)=KC0(G,L)
      KC1(G,K)=KC1(G,L)
      KC2(G,K)=KC2(G,L)
      KD0(G,K)=KD0(G,L)
      KD1(G,K)=KD1(G,L)
      KD2(G,K)=KD2(G,L)
    5 CONTINUE
      P(G,K)=P(G,L)
```

```
      P1(G,K)=P1(G,L)
      Q(G,K)=Q(G,L)
      Q1(G,K)=Q1(G,L)
      R(G,K)=R(G,L)
      PO(G,K)=PO(G,L)
      PO7(G,K)=PO7(G,L)
      PH(G,K)=PH(G,L)
      PH7(G,K)=PH7(G,L)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BHSET(K)                                               BHSE0001
C        SETS UP THE /BH/ ARRAYS FOR GIF:                               BHSE0002
      IMPLICIT REAL*8 (A-H,L-Z)                                         BHSE0003
      COMMON /BH/ X(101), H(101), Z(101)                                BHSE0004
      DO 1  I=1,K                                                       BHSE0005
    1 Z(I)=X(I)-X(1)                                                    BHSE0006
      RETURN                                                            BHSE0007
      END                                                               BHSE0008
```

```
      DOUBLE PRECISION FUNCTION GIF(N,G1,F,G2,C,G,K,ITC)
C     INTEGRATES:  F(G1,J)*C(G2,J)*G(G2,J) * (Z/H)**N
C     OVER ALL K SUBREGIONS J
C     WHERE Z RUNS FROM 0 TO X(K+1)-X(1) IN THIS REGION.
C     WHERE THE FORM OF F AND G IN REGION J IS GIVEN BY ITC:
C        ITC = 0:  F AND G ARE BOTH CONSTANT.
C        ITC = 1:  F IS LINEAR AND G IS CONSTANT.
C        ITC = 2:  F AND G ARE BOTH LINEAR.
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /BH/ X(101),H(101),Z(101)
      DIMENSION F(2,101),G(2,101),C(2,100)
      INTEGER G1,G2
      N1=N+1
      SUMJ=0.0D0
      IF (ITC.EQ.0)  GO TO 40
      IF (ITC.EQ.1)  GO TO 20
C        LINEAR F AND G IN REGIONS J:
      DO 10  J=1,K
      A=F(G1,J)*G(G2,J)
      B=F(G1,J)*G(G2,J+1)+F(G1,J+1)*G(G2,J)
      D=F(G1,J+1)*G(G2,J+1)
      SUML=0.0D0
      DO 5  I=1,N1
      L=I-1
      M=N-L
      IF (H(J).EQ.0.0.AND.L.EQ.0)  GO TO 1
      EH=H(J)**L
      GO TO 2
    1 EH=1.0
    2 IF (Z(J).EQ.0.0.AND.M.EQ.0)  GO TO 3
      EZ=Z(J)**M
      GO TO 4
    3 EZ=1.0
    4 SUML=SUML+FACT(N)/(FACT(M)*FACT(L))*EH*EZ*
     X   (2.D0*A/DFLOAT((L+3)*(L+2)*(L+1))+B/DFLOAT((L+3)*(L+2))
     X   +D/DFLOAT(L+3))
```

```
    5 CONTINUE
      SUMJ=SUMJ+H(J)*C(G2,J)*SUML
   10 CONTINUE
      GO TO 100
C        LINEAR F AND CONSTANT G IN REGIONS J:
   20 DO 30  J=1,K
      SUML=0.0D0
      DO 25  I=1,N1
      L=I-1
      M=N-L
      IF (H(J).EQ.0.0.AND.L.EQ.0)  GO TO 21
      EH=H(J)**L
      GO TO 22
   21 EH=1.0
   22 IF (Z(J).EQ.0.0.AND.M.EQ.0)  GO TO 23
      EZ=Z(J)**M
      GO TO 24
   23 EZ=1.0
   24 SUML=SUML+FACT(N)/(FACT(M)*FACT(L))*EH*EZ*
     X   (F(G1,J)/DFLOAT((L+1)*(L+2))+F(G1,J+1)/DFLOAT(L+2))
   25 CONTINUE
      SUMJ=SUMJ+H(J)*C(G2,J)*G(G2,J)*SUML
   30 CONTINUE
      GO TO 100
C        CONSTANT F AND G IN REGIONS J:
   40 DO 50  J=1,K
      SUML=0.0D0
      DO 55  I=1,N1
      L=I-1
      M=N-L
      IF (H(J).EQ.0.0.AND.L.EQ.0)  GO TO 51
      EH=H(J)**L
      GO TO 52
   51 EH=1.0
   52 IF (Z(J).EQ.0.0.AND.M.EQ.0)  GO TO 53
      EZ=Z(J)**M
```

```
      GO TO 54                                                       GIF 0073
   53 EZ=1.0                                                         GIF 0074
   54 SUML=SUML+FACT(N)/(FACT(M)*FACT(L))*EH*EZ*                     GIF 0075
     X    (1./DFLOAT(L+1))                                           GIF 0076
   55 CONTINUE                                                       GIF 0077
      SUMJ=SUMJ+H(J)*F(G1,J)*C(G2,J)*G(G2,J)*SUML                    GIF 0078
   50 CONTINUE                                                       GIF 0079
  100 GIF=SUMJ/(X(K+1)-X(1))**N                                      GIF 0080
      RETURN                                                         GIF 0081
      END                                                            GIF 0082
```

```
      DOUBLE PRECISION FUNCTION FACT(N)                                 FACT0001
C     COMPUTES N FACTORIAL:                                             FACT0002
      FACT=1.0D0                                                        FACT0003
      IF (N.LE.1)  RETURN                                               FACT0004
      DO 1  I=2,N                                                       FACT0005
    1 FACT=FACT*DFLOAT(I)                                               FACT0006
      RETURN                                                            FACT0007
      END                                                               FACT0008
```

```
      SUBROUTINE PRTOUT(IP)
C         IP = 1:  PRINTS OUT THE /B5/ ARRAYS.
C         IP = 2:  PRINTS OUT THE /B3/ MATRICES GIVEN TO POWER.
      IMPLICIT REAL*8 (A-H,K-Z)
      COMMON /B2/ KR, N
      COMMON /B3/ L1(26,26),L2(26,26),F1(26,26),F4(26,26),
     X            F2(26,26),F3(26,26),T(26,26)
      COMMON /B5/  KA0(2,25),KA1(2,25),KA2(2,25),KB0(2,25),KB1(2,25),
     X             KB2(2,25),LA0(2,25),LA1(2,25),LA2(2,25),SR0(1,25),
     X             SR1(1,25),SR2(1,25),KC0(1,25),KC1(1,25),KC2(1,25),
     X             KD0(1,25),KD1(1,25),KD2(1,25),
     X             P(2,25),P1(2,25),Q(2,25),
     X             Q1(2,25), R(2,25),  P0(2,25), P07(2,25),PH(2,25),
     X             PH7(2,25)
      COMMON /XAXIS/ HX,HR(25)
      INTEGER KR, G, N
      GO TO (1001,1002),IP
C         KA AND KB ARRAYS:
 1001 WRITE (6,10)
   10 FORMAT ('1    G',4X,'I',12X,'KA0(G,I)',12X,'KA1(G,I)',12X,
     X   'KA2(G,I)',12X,'KB0(G,I)',12X,'KB1(G,I)',12X,'KB2(G,I)')
      DO 11  G=1,2
      WRITE (6,12)
   11 WRITE (6,15) (G,I,KA0(G,I),KA1(G,I),KA2(G,I),KB0(G,I),KB1(G,I),
     X              KB2(G,I),I=1,KR)
   12 FORMAT (' ')
   15 FORMAT (2I5,6D20.7)
C         KC AND KD ARRAYS:
      WRITE (6,16)
   16 FORMAT ('1    G',4X,'I',12X,'KC0(G,I)',12X,'KC1(G,I)',12X,
     X   'KC2(G,I)',12X,'KD0(G,I)',12X,'KD1(G,I)',12X,'KD2(G,I)')
      G=1
      WRITE (6,12)
   17 WRITE (6,15) (G,I,KC0(G,I),KC1(G,I),KC2(G,I),KD0(G,I),KD1(G,I),
     X              KD2(G,I),I=1,KR)
C         LA AND SR ARRAYS:
```

```
      WRITE (6,20)
   20 FORMAT ('1   G',4X,'I',12X,'LA0(G,I)',12X,'LA1(G,I)',12X,
     X   'LA2(G,I)',12X,'SR0(G,I)',12X,'SR1(G,I)',12X,'SR2(G,I)')
      G=1
      WRITE (6,12)
      WRITE (6,15) (G,I,LA0(G,I),LA1(G,I),LA2(G,I),SR0(G,I),SR1(G,I),
     X              SR2(G,I),I=1,KR)
      G=2
      WRITE (6,12)
      WRITE (6,25) (G,I,LA0(G,I),LA1(G,I),LA2(G,I),I=1,KR)
   25 FORMAT (2I5,3D20.7)
C        P, Q, AND R ARRAYS:
      WRITE (6,30)
   30 FORMAT ('1   G',4X,'I',14X,'P(G,I)',13X,'P1(G,I)',14X,'Q(G,I)',
     X   13X,'Q1(G,I)',14X,'R(G,I)')
      DO 31  G=1,2
      WRITE (6,12)
   31 WRITE (6,35) (G,I,P(G,I),P1(G,I),Q(G,I),Q1(G,I),R(G,I),I=1,KR)
   35 FORMAT (2I5,5D20.7)
C        P0, PH, AND HR ARRAYS:
      WRITE (6,40)
   40 FORMAT ('1   G',4X,'I',13X,'P0(G,I)',12X,'P07(G,I)',13X,'PH(G,I)',
     X   12X,'PH7(G,I)',15X,'HR(I)')
      DO 41  G=1,2
      WRITE (6,12)
   41 WRITE (6,45) (G,I,P0(G,I),P07(G,I),PH(G,I),PH7(G,I),HR(I),I=1,KR)
   45 FORMAT (2I5,5D20.7)
      GO TO 100
C        PRINT OUT THE /B3/ MATRICES:
 1002 WRITE (6,50)
   50 FORMAT ('1MATRIX L1:',/)
      DO 51  I=1,N
   51 WRITE (6,55) (L1(I,J),J=1,N)
   55 FORMAT (10D12.3,/,(2X,10D12.3))
      WRITE (6,60)
   60 FORMAT ('1MATRIX L2:',/)
```

```
      DO 61  I=1,N
   61 WRITE (6,55) (L2(I,J),J=1,N)
      WRITE (6,70)
   70 FORMAT ('1MATRIX F1:',/)
      DO 71  I=1,N
   71 WRITE (6,55) (F1(I,J),J=1,N)
      WRITE (6,80)
   80 FORMAT ('1MATRIX F2:',/)
      DO 81  I=1,N
   81 WRITE (6,55) (F2(I,J),J=1,N)
      WRITE (6,82)
   82 FORMAT ('1MATRIX F3:',/)
      DO 83  I=1,N
   83 WRITE (6,55) (F3(I,J),J=1,N)
      WRITE (6,84)
   84 FORMAT ('1MARTIX F4:',/)
      DO 85  I=1,N
   85 WRITE (6,55) (F4(I,J),J=1,N)
      WRITE (6,90)
   90 FORMAT ('1MATRIX T:',/)
      DO 91  I=1,N
   91 WRITE (6,55) (T(I,J),J=1,N)
  100 RETURN
      END
```

```
      SUBROUTINE POWER
C     SOLVES THE 2*N MULTIGROUP EQUATIONS:  M*PHI = (1/LAMDA)*F*PHI
C     BY THE FISSION SOURCE POWER METHOD
C     USING SIMULTANEOUS OVERRELAXATION.
C        WHERE:  M AND F ARE DOUBLE PRECISION 2N BY 2N BLOCK MATRICES;
C        AND:    PHI IS THE 2N FLUX (FAST AND THERMAL) VECTOR.
C                   L1*PHI1 = CHI1*(F1*PHI1 + F2*PHI2)
C        -T*PHI1 + L2*PHI2 = CHI2*(F3*PHI1 + F4*PHI2)
C        METHOD FOLLOWS WACHPRESS, PAGE 83.  SOLUTION BY GROUP ITERATION.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH,ISEE
      COMMON /B2/ KR,N
      COMMON /B3/ L1(26,26),L2(26,26),F1(26,26),F4(26,26),
     X            F2(26,26),F3(26,26),T(26,26)
      COMMON /B4/ PHI(2,26), PSI(2,26), LAMDA, ICOUT
      COMMON /B5/ S(26), ERROR(2,26), Z(26)
      COMMON /B6/ TE1(2,5),TE2(2,5),TE3(5),IN(5)
      COMMON /CHIF/ CHI(2)
      COMMON /XAXIS/ HX,HR(25)
      COMMON /ER/ EPS1,EPS2,EPS3
      COMMON /FSTR/ PHISTR(2,26,6)
      COMMON /ESTR/ LAMSTR(300), EFSTR(2,300), EFMSTR(2,300), ERLAM(300)
      COMMON /TRUE/ TRULAM, TRUPHI(2,26), PHICON(2,300), LAMCON(300),IFT
      DIMENSION PSI1(26), PSI2(26), SQ(2), DPHI(2), ERRMAX(2)
      INTEGER N
C        DEFAULT OPTIONS FOR THE TRUE EIGENVALUE AND FLUXES:
      TRULAM=1.0
      DO 5  IG=1,2
      DO 5  I=1,N
    5 TRUPHI(IG,I)=1.0
C        DEFAULT OPTIONS FOR POWER PARAMETERS:
      ALPHA=1.25
      LAMDA=1.0
      DO 555  IG=1,2
      IF (IBC.NE.4)  GO TO 551
      DO 550  I=1,N
```

```
  550 PHI(IG,I)=1.0                                                          POWE0037
      GO TO 555                                                              POWE0038
  551 X=3.1415926/HX                                                         POWE0039
      IF (IBC.NE.1)  X=X/2.0                                                 POWE0040
      SUM1=0.0                                                               POWE0041
      DO 552  K=1,KR                                                         POWE0042
      SUM1=SUM1+HR(K)                                                        POWE0043
  552 PHI(IG,K)=DSIN(SUM1*X)                                                 POWE0044
  555 CONTINUE                                                               POWE0045
C        READ IN THE TRUE (EXPECTED) EIGENVALUE AND FLUX VECTOR (MINUS 0 BC'S):  POWE0046
      IFT=0                                                                  POWE0047
      READ (5,500,END=501) TRULAM, (TRUPHI(1,I),I=1,N)                       POWE0048
      READ (5,503,END=501) (TRUPHI(2,I),I=1,N)                               POWE0049
      IFT=1                                                                  POWE0050
  500 FORMAT (E25.14,/,(4E20.10))                                            POWE0051
C        READ IN:  OVERRELAXATION PARAMETERS  ;  ALPHA  (OUTER ITERATION)    POWE0052
C                  INITIAL GUESS AT EIGENVALUE;  LAMDA                       POWE0053
C                  INITIAL NORMALIZED FLUX    ;  PHI(1-N)                    POWE0054
  501 READ (5,506,END=510) ALPHA                                             POWE0055
      READ (5,502,END=510) LAMDA                                             POWE0056
      READ (5,503) (PHI(1,I),I=1,N)                                          POWE0057
      READ (5,503) (PHI(2,I),I=1,N)                                          POWE0058
  506 FORMAT (F10.5)                                                         POWE0059
  502 FORMAT (E25.14)                                                        POWE0060
  503 FORMAT ((4E20.10))                                                     POWE0061
  510 CONTINUE                                                               POWE0062
C        STORING FOR PRINTING THE MULTIGRCUP FLUX SHAPE.                     POWE0063
      DO 11  IG=1,2                                                          POWE0064
      DO 10  I=1,N                                                           POWE0065
   10 PHISTR(IG,I,2)=PHI(IG,I)                                               POWE0066
C        FILL RUNNING COORD IN PHISTR                                        POWE0067
      KR1=KR+1                                                               POWE0068
      DO 11  I=1,KR1                                                         POWE0069
   11 PHISTR(IG,I,1)=DFLOAT(I)                                               POWE0070
C        IK IS THE FLUX PLOTTING CCUNTER.                                    POWE0071
      IK=1                                                                   POWE0072
```

```
C          STORES THE ITERATION NUMBER FOR FLUX HISTORY PLOTTING:
      IN(1)=0
C          STORES TEMPORARY ERRORS FOR FLUX HISTORY PLOTTING:
      TE1(1,1)=0.
      TE1(2,1)=0.
      TE2(1,1)=0.
      TE2(2,1)=0.
      TE3(1)=0.0
C          EIGENVALUE OF THE PREVIOUS ITERATION:
      LAMB4=LAMDA
C          THE MAXIMUM NUMBER OF ALLOWED ITERATIONS: ICMAX
      ICMAX=300
C          PRINT OUT THE POWER METHOD PARAMETER INFORMATION:
      WRITE (6,700) ICMAX,ALPHA,LAMDA,(PHI(1,I),I=1,N)
      WRITE (6,701) (PHI(2,I),I=1,N)
  700 FORMAT ('1EXECUTING MULTIGROUP FISSION SOURCE POWER ITERATION METH
     XOD.',///,
     X    5X,'MAXIMUM NUMBER OF ALLOWABLE ITERATIONS:',/,
     X    10X,'ICMAX =',I4,///,
     X    5X,'OUTER ITERATION RELAXATION PARAMETER:',/,
     X    10X,'ALPHA =',F7.3,//,
     X    5X,'INITIAL GUESS AT EIGENVALUE:',/,
     X    10X,'LAMBDA =',E22.14,//,
     X    5X,'INITIAL GUESS AT THE GROUP FLUX SHAPE CONNECTION POINTS:',
     X    //,8X,'FAST GROUP:',/,
     X    10X,'F(K)''S =',4E25.14,/,(18X,4E25.14))
  701 FORMAT ('0',7X,'THERMAL GROUP:',/,
     X    10X,'F(K)''S =',4E25.14,/,(18X,4E25.14))
C          BEGIN ITERATION LOOP.
      ICOUT=0
C          ICOUT IS THE OUTER ITERATION COUNTER.
   20 ICOUT=ICOUT+1
      IF (ICOUT.GT.ICMAX)  GO TO 100
C          SOLVE FOR THE NEW GROUP FLUX VECTORS: PSI:
C          FAST GROUP; SOURCE VECTOR:
      DO 25  I=1,N
```

```
      S(I)=0.0                                                          POWE0109
      DO 24  J=1,N                                                      POWE0110
   24 S(I)=S(I)+F1(I,J)*PHI(1,J)+F2(I,J)*PHI(2,J)                       POWE0111
   25 S(I)=CHI(1)*S(I)                                                  POWE0112
C        FAST FLUX:                                                     POWE0113
      CALL SOLV3D(N,L1,PSI1,S)                                          POWE0114
C        THERMAL GROUP; SOURCE VECTOR:                                  POWE0115
      DO 27  I=1,N                                                      POWE0116
      S(I)=0.0                                                          POWE0117
      Z(I)=0.0                                                          POWE0118
      DO 26  J=1,N                                                      POWE0119
      S(I)=S(I)+F3(I,J)*PSI1(J)+F4(I,J)*PHI(2,J)                        POWE0120
   26 Z(I)=Z(I)+T(I,J)*PSI1(J)                                          POWE0121
   27 Z(I)=Z(I)+CHI(2)*S(I)                                             POWE0122
C        THERMAL FLUX:                                                  POWE0123
      CALL SOLV3D(N,L2,PSI2,Z)                                          POWE0124
C        CALCULATION OF THE EIGENVALUE:                                 POWE0125
      SUM1=0.0D0                                                        POWE0126
      SUM2=0.0D0                                                        POWE0127
      DO 28  I=1,N                                                      POWE0128
      SUM2=SUM2+PSI1(I)*PSI1(I)+PSI2(I)*PSI2(I)                         POWE0129
   28 SUM1=SUM1+PSI1(I)*PHI(1,I)+PSI2(I)*PHI(2,I)                       POWE0130
      LAMDA=SUM2/SUM1                                                   POWE0131
      LAMSTR(ICOUT)=LAMDA                                               POWE0132
      ERRLAM=DABS(LAMDA-LAMB4)                                          POWE0133
C        PUT PSI1 AND PSI2 INTO BIGGER PSI:                             POWE0134
      DO 30  I=1,N                                                      POWE0135
      PSI(1,I)=PSI1(I)                                                  POWE0136
   30 PSI(2,I)=PSI2(I)                                                  POWE0137
C        POINT BY POINT SIMULTANEOUS RELAXATION FLUX ITERATION:         POWE0138
      X=ALPHA                                                           POWE0139
C        DO NOT RELAX DURING THE FIRST THREE ITERATIONS:                POWE0140
      IF (ICOUT.LE.3)  X=1.0                                            POWE0141
C        CALCULATE THE NEW GROUP FLUX ITERATES AND GROUP ERRORS:        POWE0142
      DO 40  IG=1,2                                                     POWE0143
      DO 40  I=1,N                                                      POWE0144
```

```
   40 PSI(IG,I)=PHI(IG,I)+X*(PSI(IG,I)/LAMDA-PHI(IG,I))
      CALL NORMAL(PSI,N)
      DO 39  IG=1,2
      ERRMAX(IG)=0.0
      SQ(IG)=0.0
      DO 38  I=1,N
      ERROR(IG,I)=DABS((PSI(IG,I)-PHI(IG,I))/PSI(IG,I))
      IF (ERROR(IG,I).GT.ERRMAX(IG))  ERRMAX(IG)=ERROR(IG,I)
      SQ(IG)=SQ(IG)+ERROR(IG,I)**2
C        UPDATE THE FLUX ITERATE:
   38 PHI(IG,I)=PSI(IG,I)
   39 SQ(IG)=DSQRT(SQ(IG))
C        NORMALIZE PSI GROUPS TO UNITY:
      CALL NORM2(PSI,TRUPHI,N)
      IF (IFT.EQ.0)  GO TO 37
      DLAM=LAMDA-TRULAM
      DO 36  IG=1,2
      DPHI(IG)=0.0
      DO 35  I=1,N
   35 DPHI(IG)=DPHI(IG)+(PSI(IG,I)-TRUPHI(IG,I))**2
   36 DPHI(IG)=DSQRT(DPHI(IG))
   37 IF (IPLOT.NE.2)  GO TO 45
C        THE FOLLJWING IS FOR NICELY PLOTTING THE GROUP FLUX HISTORY.
         DO 41  IG=1,2
         DO 41  I=1,N
   41    ERROR(IG,I)=PSI(IG,I)
C           ERROR NOW CONTAINS THE NEW NORMALIZED FLUX ITERATE PHI.
         JK=IK
         IF (IK.EQ.0)  JK=5
         DO 42  IG=1,2
         DO 42  I=1,N
         IF (DABS(ERROR(IG,I)-PHISTR(IG,I,JK+1)).GE.0.01)  GO TO 43
   42    CONTINUE
C           FLUX HAS NOT CHANGED ENOUGH FOR PLOTTING.
         GO TO 45
C           SAVE THE NORMALIZED FLUX FOR PLOTTING:
```

```
   43     IK=IK+1                                                              POWE0181
          IN(IK)=ICOUT                                                         POWE0182
          TE3(IK)=ERRLAM                                                       POWE0183
          DO 44  IG=1,2                                                        POWE0184
          TE1(IG,IK)=ERRMAX(IG)                                                POWE0185
          TE2(IG,IK)=SQ(IG)                                                    POWE0186
          DO 44  I=1,N                                                         POWE0187
   44     PHISTR(IG,I,IK+1)=ERROR(IG,I)                                        POWE0188
          IF (IK.NE.5)  GO TO 45                                               POWE0189
C            PLOT THE LAST FIVE SAVED FLUXES:                                  POWE0190
          CALL PHIPLT(5)                                                       POWE0191
          IK=0                                                                 POWE0192
   45 CONTINUE                                                                 POWE0193
C         ERROR CRITERIA FOR ACCEPTANCE OF CONVERGENCE.                        POWE0194
      IFLAG1=0                                                                 POWE0195
      IFLAG2=0                                                                 POWE0196
      IFLAG3=0                                                                 POWE0197
C         STORE THE ERRORS FOR COMPARISON:                                     POWE0198
C              ERROR BETWEEN ITERATION EIGENVALUES:                            POWE0199
      ERLAM(ICOUT)=ERRLAM                                                      POWE0200
      DO 46  IG=1,2                                                            POWE0201
C              MAXIMUM ERROR BETWEEN ITERATION FLUXES:                         POWE0202
      EFSTR(IG,ICOUT)=ERRMAX(IG)                                               POWE0203
C              MEAN SQUARE ERROR BETWEEN ITERATION FLUXES:                     POWE0204
      EFMSTR(IG,ICOUT)=SQ(IG)                                                  POWE0205
C              MEAN SQUARE ERROR BETWEEN THE ITERATION FLUX AND GIVEN TRUE FLUX:  POWE0206
      PHICON(IG,ICOUT)=DPHI(IG)                                                POWE0207
   46 CONTINUE                                                                 POWE0208
C              ERROR BETWEEN THE ITERATION EIGENVALUE AND GIVEN TRUE EIGENVALUE:  POWE0209
      LAMCON(ICOUT)=DLAM                                                       POWE0210
      IF ((ERRMAX(1).LT.EPS1).AND.(ERRMAX(2).LT.EPS1))    IFLAG1=1             POWE0211
      IF ((SQ(1).LT.EPS2).AND.(SQ(2).LT.EPS2))            IFLAG2=1             POWE0212
      IF (ERRLAM.LT.EPS3)  IFLAG3=1                                            POWE0213
      IFLAG4=IFLAG1*IFLAG2*IFLAG3                                              POWE0214
      IF (IFLAG4.EQ.1)  GO TO 50                                               POWE0215
C         OTHERWISE CONTINUE THE ITERATION.                                    POWE0216
```

```
      LAMB4=LAMDA                                                       POWE0217
      GO TO 20                                                          POWE0218
   50 CONTINUE                                                          POWE0219
C        CONVERGENCE ACCOMPLISHED.                                      POWE0220
C        NORMALIZE THE CONVERGED FLUX VECTOR:                           POWE0221
      CALL NORMAL(PHI,N)                                                POWE0222
C        PLOT ANY LEFT OVER FLUX HISTORY PLOTS:                         POWE0223
      IF ((IPLOT.EQ.2).AND.(IK.NE.0))  CALL PHIPLT(IK)                  POWE0224
C        BOUNDRY CONDITION INSERTIONS.                                  POWE0225
      IER=0                                                             POWE0226
C        IER ALLOWS B.C. INSERTIONS FOR YES AND NO CONVERGENCE:         POWE0227
   55 IF (IBC.EQ.4)  GO TO 90                                           POWE0228
      IF (IBC.NE.3)  GO TO 60                                           POWE0229
      PHI(1,KR+1)=0.                                                    POWE0230
      PHI(2,KR+1)=0.                                                    POWE0231
      GO TO 90                                                          POWE0232
   60 DO 70  I=1,N                                                      POWE0233
      J=N+1-I                                                           POWE0234
      PHI(1,J+1)=PHI(1,J)                                               POWE0235
   70 PHI(2,J+1)=PHI(2,J)                                               POWE0236
      IF (IBC.EQ.5.OR.IBC.EQ.7)  GO TO 71                               POWE0237
      PHI(1,1)=0.0                                                      POWE0238
      PHI(2,1)=0.0                                                      POWE0239
      GO TO 72                                                          POWE0240
   71 PHI(1,1)=PHI(1,2)                                                 POWE0241
      PHI(2,1)=PHI(2,2)                                                 POWE0242
   72 IF (IBC.NE.1)  GO TO 73                                           POWE0243
      PHI(1,KR+1)=0.0                                                   POWE0244
      PHI(2,KR+1)=0.0                                                   POWE0245
      GO TO 90                                                          POWE0246
   73 IF (IBC.LT.6)  GO TO 90                                           POWE0247
      PHI(1,KR+1)=PHI(1,KR)                                             POWE0248
      PHI(2,KR+1)=PHI(2,KR)                                             POWE0249
   90 IF (IER.EQ.1)  GO TO 102                                          POWE0250
      RETURN                                                            POWE0251
C        NO CONVERGENCE ACCOMPLISHED:                                   POWE0252
```

```
  100 CONTINUE                                                               POWE0253
C         NORMALIZE THE UNCONVERGED FLUX:                                    POWE0254
      CALL NORMAL(PHI,N)                                                     POWE0255
      ICOUT=ICOUT-1                                                          POWE0256
      WRITE (6,101) ICOUT                                                    POWE0257
  101 FORMAT (1H1,'POWER METHOD DID NOT CONVERGE FOR THIS CASE AFTER',       POWE0258
     X   I4,' ITERATIONS.',//,1X,'EXECUTION TERMINATED ')                    POWE0259
      IER=1                                                                  POWE0260
      GO TO 55                                                               POWE0261
  102 CONTINUE                                                               POWE0262
C         FOR PRINTING OUT THE EIGENVALUE HISTORY AND THE FINAL FLUX SHAPE:  POWE0263
      IPLOT=1                                                                POWE0264
      JPLOT=1                                                                POWE0265
      RETURN                                                                 POWE0266
      END                                                                    POWE0267
```

```
      SUBROUTINE SOLV3D(N,A,X,Y)
C     SOLVES THE N DOUBLE PRECISION MATRIX EQUATIONS:  A*X = Y,
C     FOR X - GIVEN THE N BY N TRIDIAGONAL MATRIX A
C     AND THE SOURCE VECTOR Y.
C        METHOD IS FORWARD ELIMINATION FOLLOWED BY BACKWARD SUBSTITUTION.
C        CF - WACHPRESS, PAGE 23.
      REAL*8 A, X, Y, H, P, D
      DIMENSION A(26,26), X(26), Y(26), H(26), P(26)
      IF (A(1,1).EQ.0.0)  GO TO 10
      H(1)=-A(1,2)/A(1,1)
      P(1)=Y(1)/A(1,1)
      DO 1  M=2,N
      D=A(M,M)+A(M,M-1)*H(M-1)
      IF (D.EQ.0.0)  GO TO 20
      P(M)=(Y(M)-A(M,M-1)*P(M-1))/D
      IF (M.EQ.N)  GO TO 1
      H(M)=-A(M,M+1)/D
    1 CONTINUE
      X(N)=P(N)
      DO 2  I=2,N
      M=N+1-I
    2 X(M)=P(M)+H(M)*X(M+1)
      RETURN
C        IN CASE OF ANY IMPENDING ZERO DIVISORS:
   10 WRITE (6,11)
   11 FORMAT ('0FIRST ELEMENT OF A, A(1,1), IS ZERO.',/,
     X    5X,'BETTER FIX IT BOSS.')
      GO TO 30
   20 WRITE (6,21) M
   21 FORMAT ('0ZERO DIVISOR ENCOUNTERED IN EQUATION M =',I3,'.',/,
     X    5X,'BETTER FIX IT BOSS.')
   30 WRITE (6,31)
   31 FORMAT ('0EXECUTION TERMINATED.')
      CALL EXIT
      RETURN
      END
```

```
      SUBROUTINE NORMAL(PHI,N)
C     NORMALIZES THE GROUP FLUXES TO ONE. NOT BOTH GROUPS.
      REAL*8 PHI(2,26), A
      A=DABS(PHI(1,1))
      DO 1  IG=1,2
      DO 1  I=1,N
      IF (DABS(PHI(IG,I)).GT.A)  A=DABS(PHI(IG,I))
    1 CONTINUE
      DO 2  IG=1,2
      DO 2  I=1,N
    2 PHI(IG,I)=PHI(IG,I)/A
      RETURN
      END
```

```
      SUBROUTINE PHIPLT(L)
C        PLOTS THE GROUP FLUX HISTORY, WITH UP TO 5 GROUP FLUXES PER PLOT.
C        FAST AND THERMAL GROUP FLUXES ARE PLOTTED SEPERATELY.
C        L IS THE NUMBER OF FLUXES TO BE PLOTTED.
C        L IS BETWEEN 1 AND 5.
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /B1/ IBC
      COMMON /B2/ KR,N
      COMMON /B5/ S(26), A(26,6), B(26,6)
      COMMON /B6/ TE1(2,5),TE2(2,5),TE3(5),IN(5)
      COMMON /ER/ EPS1,EPS2,EPS3
      COMMON /FSTR/ PHISTR(2,26,6)
      DIMENSION SYMBOL(5)
      INTEGER SYMBOL /'.','-','+','#','*'/
      KR1=KR+1
C        SET UP B.C. CONDITIONS
      IF (IBC.EQ.4)  GO TO 5
      IF (IBC.EQ.3)  GO TO 3
      DO 2  IG=1,2
      DO 2  K=1,L
      DO 1  I=1,N
      J=N+1-I
    1 PHISTR(IG,J+1,K+1)=PHISTR(IG,J,K+1)
    2 PHISTR(IG,1,K+1)=0.
    3 IF (IBC.EQ.2)  GO TO 5
      DO 4  IG=1,2
      DO 4  K=1,L
    4 PHISTR(IG,KR1,K+1)=0.
    5 CONTINUE
C        FLUXES IN PHISTR HAVE BEEN NORMALIZED IN POWER.
C        PUT THE FAST FLUX IN A, AND THE THERMAL FLUX IN B:
      L1=L+1
      DO 10  K=1,L1
      DO 10  I=1,KR1
      A(I,K)=PHISTR(1,I,K)
   10 B(I,K)=PHISTR(2,I,K)
```

```
C         PLOT THE L FAST FLUX SHAPES ON ONE GRAPH:
      CALL PRTPLT(0,A,KR1,L1,KR1,0,26,6,2)
      WRITE (6,20)
   20 FORMAT (/,'0FAST FLUX ITERATION HISTORY PLOT.',/)
      WRITE (6,30)
   30 FORMAT (
     X   '0KEY:',5X,'SYMBOL',5X,'ITERATION NUMBER:',7X,'ERROR CRITERIA',
     X   11X,'ERROR',13X,'TOLERANCE')
      DO 35  I=1,L
   35 WRITE (6,40) SYMBOL(I),IN(I),TE1(1,I),EPS1,TE2(1,I),EPS2,
     X              TE3(I),EPS3
   40 FORMAT (/,12X,A1,15X,I3,16X,'FLUX',14X,1PD15.5,5X,1PD15.5,/,
     X   47X,'MEAN SQ. FLUX',5X,1PD15.5,5X,1PD15.5,/,
     X   47X,'EIGENVALUE',8X,1PD15.5,5X,1PD15.5)
C         PLOT THE L THERMAL FLUX SHAPES ON THE OTHER GRAPH:
      CALL PRTPLT(0,B,KR1,L1,KR1,0,26,6,2)
      WRITE (6,50)
   50 FORMAT (/,'0THERMAL FLUX ITERATION PLOT.',/)
      WRITE (6,30)
      DO 55  I=1,L
   55 WRITE (6,40) SYMBOL(I),IN(I),TE1(2,I),EPS1,TE2(2,I),EPS2,
     X              TE3(I),EPS3
      RETURN
      END
```

```
      SUBROUTINE OUTPUT
C        PRINTS THE RESULTS OF THE METHOD.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH
      COMMON /B2/ KR,N
      COMMON /B4/ PHI(2,26), PSI(2,26), LAMDA, ICOUT
      COMMON /ER/ EPS1,EPS2,EPS3
      COMMON /ESTR/ LAMSTR(300),EFSTR(2,300),EFMSTR(2,300),ERLAM(300)
      COMMON /TRUE/ TRULAM, TRUPHI(2,26), PHICON(2,300), LAMCON(300),IFT
      INTEGER N
      KR0=KR-1
      KR1=KR+1
      WRITE (6,1)
    1 FORMAT ('1RESULTS OF THE MULTIGROUP METHOD:')
      WRITE (6,10) ICOUT
   10 FORMAT (//,' PROBLEM TERMINATED AFTER',I5,
     X        ' OUTER (POWER) ITERATIONS TO:')
      WRITE (6,20) LAMDA
   20 FORMAT (/,10X,'LAMDA = ',1PE21.14)
C        PRINT OUT EIGENVALUES.
      CALL PLOT
      WRITE (6,30)
   30 FORMAT ('1RESULTS AFTER PROBLEM TERMINATION:',/,
     X  'ONUMBER',5X,'THERMAL FLUX POINTS',5X,'FAST FLUX POINTS')
      WRITE (6,50) (K,PHI(2,K),PHI(1,K),K=1,KR1)
   50 FORMAT (I5,1PE26.7,1PE21.7)
      IF (IPUNCH.EQ.1)  CALL PUNCH
C        CALCULATE THE FINAL TO EXPECTED FLUX RATIOS:
C        NORMALIZE BOTH PHI GROUP FLUXES FOR TRUPHI COMPARISON:
      CALL NORM2(PHI,TRUPHI,KR1)
      K1=1
      K2=KR1
      IF (IBC.LE.2)  K1=2
      IF ((IBC.EQ.1).OR.(IBC.EQ.3))  K2=KR
      DO 60  IG=1,2
      IF (IBC.LE.2)  PSI(IG,1)=1.0
```

```
      IF ((IBC.EQ.1).OR.(IBC.EQ.3))  PSI(IG,KR1) = 1.0
      I=0
      DO 60  K=K1,K2
      I=I+1
   60 PSI(IG,K)=PHI(IG,K)/TRUPHI(IG,I)
      WRITE (6,70) (I,PSI(2,I),PSI(1,I),I=1,KR1)
   70 FORMAT ('1RATIOS OF THE TERMINATED GROUP FLUX TO THE EXPECTED GROU
     XP FLUX:',//,
     X    10X,'- AN INDICATION OF THE ACCURACY OF THE CONVERGENCE -',///,
     X    '     K',12X,'THERMAL RATIO',15X,'FAST RATIO',//,(I5,2E25.10))
C         PRINT OUT THE STORED ITERATION ERRORS:
      WRITE (6,110) EPS1,(EFSTR(2,I),I=1,ICOUT)
      WRITE (6,111) EPS1,(EFSTR(1,I),I=1,ICOUT)
      WRITE (6,112) EPS2,(EFMSTR(2,I),I=1,ICOUT)
      WRITE (6,113) EPS3,(EFMSTR(1,I),I=1,ICOUT)
      WRITE (6,114) EPS3,(ERLAM(I),I=1,ICOUT)
  110 FORMAT ('1MAXIMUM NORMALIZED ERRORS BETWEEN THE THERMAL FLUX ITERA
     XTIONS:',
     X    25X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  111 FORMAT ('1MAXIMUM NORMALIZED ERRORS BETWEEN THE FAST FLUX ITERATIO
     XNS:',
     X    25X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  112 FORMAT ('1MEAN SQUARE NORMALIZED ERROR BETWEEN THE THERMAL FLUX IT
     XERATIONS:',
     X    18X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  113 FORMAT ('1MEAN SQUARE NORMALIZED ERROR BETWEEN THE FAST FLUX ITERA
     XTIONS:',
     X    18X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  114 FORMAT ('1ERROR BETWEEN THE ITERATION EIGENVALUES:',
     X    28X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
C         PRINT OUT THE GIVEN TRUE EIGENVALUE AND FLUX:
      IF (IFT.EQ.0)  RETURN
      WRITE (6,115) TRULAM,((TRUPHI(3-J,I),J=1,2),I=1,N)
  115 FORMAT ('1THE GIVEN TRUE EIGENVALUE:',//,15X,
     X    'TRULAM =',E22.14,///,
     X    '0THE GIVEN MULTIGROUP FLUXES:',//,
```

```
     X    13X,'THERMAL',16X,'FAST',//,(2D20.10))
C          PRINT OUT THE STORED CONVERGENCE ERRORS:
      WRITE (6,120) (PHICON(2,I),I=1,ICOUT)
      WRITE (6,121) (PHICON(1,I),I=1,ICOUT)
      WRITE (6,122) (LAMCON(I),I=1,ICOUT)
  120 FORMAT ('1MEAN SQUARE ERROR BETWEEN THE THERMAL ITERATION FLUX AND
     X THE GIVEN TRUE THERMAL FLUX:',//,(1P5E20.5))
  121 FORMAT ('1MEAN SQUARE ERROR BETWEEN THE FAST ITERATION FLUX AND TH
     XE GIVEN TRUE FAST FLUX:',//,(1P5E20.5))
  122 FORMAT ('1ERROR BETWEEN THE ITERATION EIGENVALUES AND THE GIVEN TR
     XUE EIGENVALUE:',//,(1P5E20.5))
      RETURN
      END
```

```
      SUBROUTINE PLOT
C        PLOTS OUT THE EIGENVALUE HISTORY AS A TABLE AND A GRAPH,
C        AS WELL AS PLOTTING OUT THE FINAL MULTIGROUP FLUX SHAPES.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH
      COMMON /B2/ KR
      COMMON /B4/ PHI(2,26), PSI(2,26), LAMDA, ICOUT
      COMMON /B5/ B(300,2)
      COMMON /ESTR/ LAMSTR(300)
      DIMENSION C(26,3)
C        IN ORDER TO SAVE SOME SPACE:
      EQUIVALENCE (B(1),C(1))
      WRITE (6,1) (LAMSTR(I),I=1,ICOUT)
    1 FORMAT ('0TABLE OF EIGENVALUES DURING THE POWER ITERATION:',
     X        //,(1P5E25.14))
      IF (JPLOT.EQ.0)  GO TO 20
      DO 10  I=1,ICOUT
      B(I,1)=I
   10 B(I,2)=LAMSTR(I)
      CALL PRTPLT(1,B,ICOUT,2,ICOUT,0,300,2,1)
      WRITE (6,11)
   11 FORMAT ('0PLOT OF THE EIGENVALUE HISTORY THROUGH THE ITERATIONS.')
   20 IF (IPLOT.EQ.0)  RETURN
      KR1=KR+1
      DO 30  I=1,KR1
      C(I,1)=I
      C(I,2)=PHI(1,I)
   30 C(I,3)=PHI(2,I)
      CALL PRTPLT(2,C,KR1,3,KR1,0,26,3,2)
      WRITE (6,31)
   31 FORMAT ('0FINAL CONVERGED CONNECTING FLUX POINTS; F(K).',//,
     X   5X,'FAST FLUX:     .',/,5X,'THERMAL FLUX:  -')
      RETURN
      END
```

```
      SUBROUTINE PUNCH                                                  PNCH0001
C        PUNCHES OUT INPUT AND OUTPUT DATA.                             PNCH0002
      COMMON /B2/ KR                                                    PNCH0003
      COMMON /B4/ F(2,26)                                               PNCH0004
      REAL*8 F                                                          PNCH0005
      KR1=KR+1                                                          PNCH0006
      WRITE (7,1) KR,(F(1,I),F(2,I),I=1,KR1)                            PNCH0007
    1 FORMAT (I5,/,(2E20.7))                                            PNCH0008
      WRITE (6,100)                                                     PNCH0009
  100 FORMAT (///,' THE OUTPUT HAS BEEN PUNCHED OUT ONTO CARDS ')       PNCH0010
      RETURN                                                            PNCH0011
      END                                                               PNCH0012
```

```
      SUBROUTINE NORM2(PSI,TRUPHI,N)
C     NORMALIZES BOTH ENERGY GROUPS OF PSI TO 1.0.
C     DITTO FOR TRUPHI ON THE FIRST CALL.
      REAL*8 PSI(2,26), TRUPHI(2,26), A(2)
      DATA K /0/
      K=K+1
C
      DO 1  IG=1,2
      A(IG)=DABS(PSI(IG,1))
      DO 1  I=1,N
      IF (DABS(PSI(IG,I)).GT.A(IG))  A(IG)=DABS(PSI(IG,I))
    1 CONTINUE
      DO 2  IG=1,2
      DO 2  I=1,N
    2 PSI(IG,I)=PSI(IG,I)/A(IG)
      IF (K.NE.1)  RETURN
      DO 5  IG=1,2
      A(IG)=0.
      DO 5  I=1,N
      IF (TRUPHI(IG,I).GT.A(IG))  A(IG)=TRUPHI(IG,I)
    5 CONTINUE
      DO 6  IG=1,2
      DO 6  I=1,N
    6 TRUPHI(IG,I)=TRUPHI(IG,I)/A(IG)
      RETURN
      END
```

```
      SUBROUTINE PRTPLT(NO,B,N,M,NL,NS,KX,JX,ISP)
C
C     *  IDENTICAL TO SUBROUTINE PRTPLT PREVIOUSLY LISTED IN PROGRAM REF2G.
C
      RETURN
      END
```

F.3. SOURCE LISTING of Program CUBIC

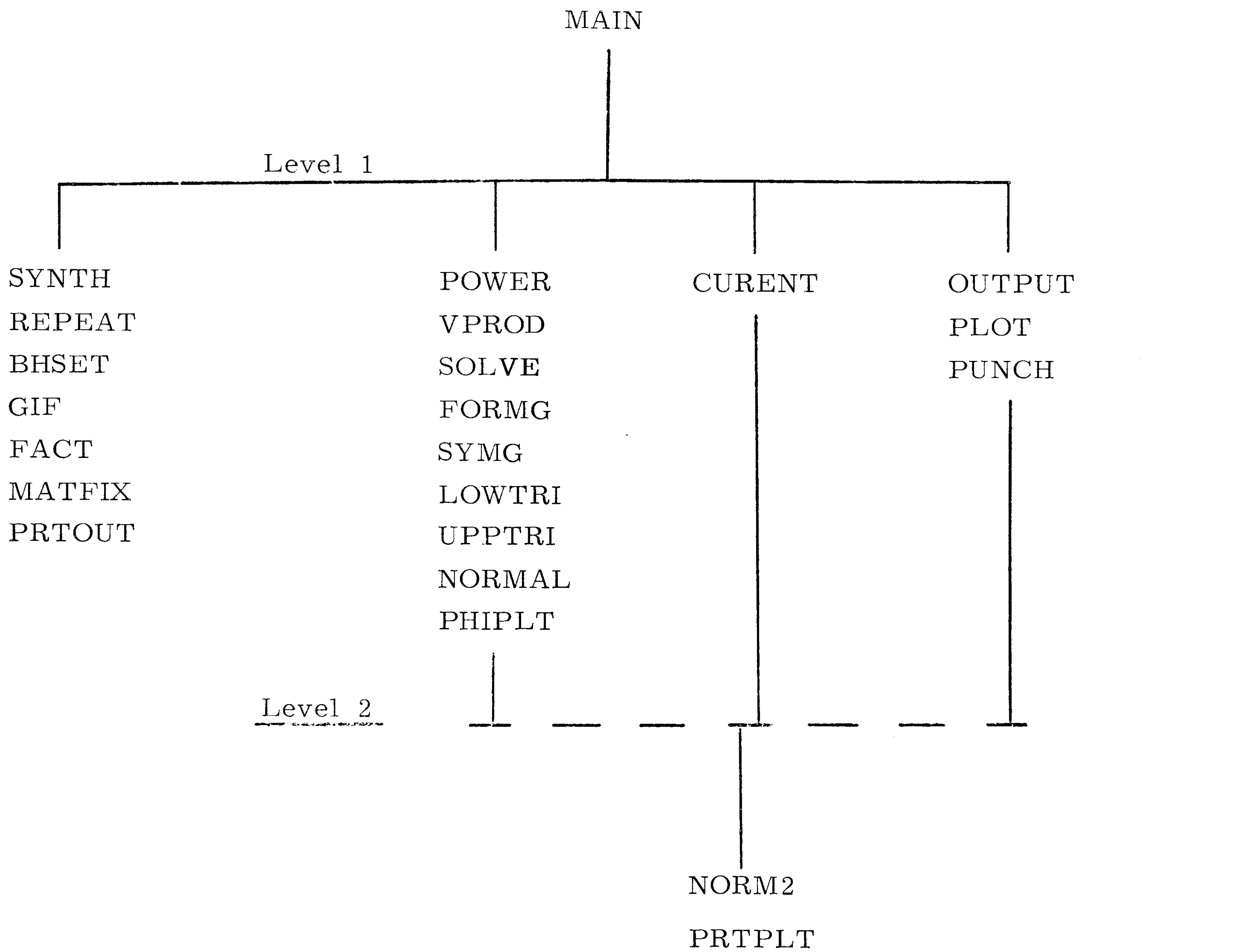


Figure F.3. Structure of Program CUBIC.

```
C     PROGRAM CUBIC:
C     TWO GROUP PROPOSED METHOD USING CUBIC HERMITE BASIS FUNCTIONS.
      CALL TIMING (I1)
      CALL SYNTH
      CALL TIMING (I4)
      CALL POWER
      CALL TIMING (I6)
      CALL CURENT
      CALL TIMING (I7)
      CALL OUTPUT
      CALL TIMING (I8)
C        TIMING EXECUTION
      WRITE (6,30)
   30 FORMAT (1H1,'TIMING PROGRAM EXECUTION:',/)
      J=I4-I1
      WRITE(6,701) J
      J=I6-I4
      WRITE(6,704) J
      J=I7-I6
      WRITE(6,706) J
      J=I8-I7
      WRITE(6,707) J
  701 FORMAT (1H ,' SYNTH HAS TAKEN',I6,' /100 SECONDS.')
  704 FORMAT (1H ,' POWER HAS TAKEN',I6,' /100 SECONDS.')
  706 FORMAT (1H ,' CURENT HAS TAKEN',I5,' /100 SECONDS.')
  707 FORMAT (1H ,' OUTPUT HAS TAKEN',I5,' /100 SECONDS.')
      CALL TIMING (I20)
      J=I20-I1
      WRITE(6,720) J
  720 FORMAT (1H0,' THIS RUN HAS TAKEN',I6,' /100 SECONDS TO RUN.')
      STOP
      END
```

```
      SUBROUTINE SYNTH
C        PROPOSED CUBIC HERMITE SYNTHESIS METHOD:
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C        ADJOINT QUANTITIES OF VARIBLES ARE DENOTED BY 7 RATHER THAN *.
C        THUS:  PHI7 (RATHER THAN PHI*) IS THE ADJOINT OF PHI. ETC.
      IMPLICIT REAL*8 (A-H,K-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH,ISEE
      COMMON /B2/ KR,NN
      COMMON /B3/ LT(50,6,2), FT(50,6,4), T(50,6)
      COMMON /B5/
     X   KA0(2,25),KA1(2,25),KA2(2,25),KA3(2,25),KA4(2,25),KA5(2,25),
     X   KA6(2,25),KB0(2,25),KB1(2,25),KB2(2,25),KB3(2,25),KB4(2,25),
     X   KB5(2,25),KB6(2,25),LA0(2,25),LA1(2,25),LA2(2,25),LA3(2,25),
     X   LA4(2,25),LA5(2,25),LA6(2,25),P0(2,25) ,P1(2,25) ,P2(2,25) ,
     X   P3(2,25) ,P4(2,25) ,P5(2,25) ,P6(2,25) ,Q0(2,25) ,Q1(2,25) ,
     X   Q2(2,25) ,Q3(2,25) ,Q4(2,25) ,Q5(2,25) ,Q6(2,25) ,R0(2,25) ,
     X   R1(2,25) ,R2(2,25) ,R3(2,25) ,R4(2,25) ,SR0(1,25),SR1(1,25),
     X   SR2(1,25),SR3(1,25),SR4(1,25),SR5(1,25),SR6(1,25),KC0(1,25),
     X   KC1(1,25),KC2(1,25),KC3(1,25),KC4(1,25),KC5(1,25),KC6(1,25),
     X   KD0(1,25),KD1(1,25),KD2(1,25),KD3(1,25),KD4(1,25),KD5(1,25),
     X   KD6(1,25),
     X   P0(2,25) ,PH(2,25) ,P07(2,25),PH7(2,25),D0(2,25) ,DH(2,25) ,
     X   C0(2), CH(2), TITLE(20), ITF(25), KTF(25)
      COMMON /CHIF/ CHI(2)
      COMMON /XAXIS/ HX, HR(25)
      COMMON /BH/ X(101),  H(101)
      COMMON /ER/ EPS1,EPS2,EPS3
      DIMENSION PHI(2,101),PHI7(2,101),CUR(2,101),CUR7(2,101)
      DIMENSION A(2,100),F(2,100),D(2,100),S(2,100),DI(2,100),XU(2,100)
C        IN ORDER TO SAVE SPACE:
      EQUIVALENCE  (XU(1),LT(1)),  (A(1),LT(201)),  (F(1),LT(401)),
     X             (DI(1),FT(1)),  (D(1),FT(201)),  (S(1),FT(401))
      REAL TITLE
      INTEGER KR,K,KS,KS1,KR0,NN,NUMITF,KTF,N
      READ (5,200) TITLE
  200 FORMAT (20A4)
```

```
      WRITE (6,201) TITLE
  201 FORMAT (1H1,20A4,//)
C        READ IN THE NUMBER OF REGION TRIAL FUNCTIONS AND TYPE OF B.C.S.
C        AS WELL AS THE TOLERANCES AND THE OUTPUT TYPES DESIRED:
      READ (5,1) KR,IBC,EPS1,EPS2,EPS3,IPLCT,JPLOT,IPUNCH,ISEE,ITW,ITC
    1 FORMAT (2I5,3D10.3,6I5)
      IF (IBC.EQ.3)  IBC=2
C        READ IN THE TYPE-NUMBER OF EACH TF REGION:
      READ (5,100) (ITF(I),I=1,KR)
  100 FORMAT (25I2)
C        READ IN THE FISSION YEILDS FOR EACH GROUP:
C        AND THE MATRIX NORMALIZATION PARAMETER:  THETA  (DEFAULT = 1.0):
      READ (5,101) CHI(1), CHI(2),  THETA
  101 FORMAT (3F10.5)
      IF (THETA.EQ.0.0)  THETA=1.0
      KR0=KR-1
      WRITE (6,2) KR, IBC, ISEE, ITW, ITC
    2 FORMAT ('0ONE DIMENSIONAL TWO GROUP CUBIC SYNTHESIS PROGRAM:',//,
     X   5X,'NUMBER OF COARSE MESH REGIONS:  KR = ',I2,/,
     X   5X,'BOUNDARY CONDITION NUMBER:     IBC = ',I2,/,
     X   5X,'AMOUNT OF OUTPUT REQUESTED:   ISEE = ',I2,//,
     X   5X,'TYPE OF WEIGHTING FUNCTIONS:   ITW = ',I2,/,
     X   5X,'TYPE OF CURRENT FUNCTIONS:     ITC = ',I2,//,
     X   5X,'REGIONAL INPUT MATERIAL PROPERTIES AND FLUX SHAPES FOLLOW',
     X   /,5X,'IF ISEE > 0:',//,
     X   5X,'FLUX SHAPES ARE LINEAR IN EACH INDICATED SUBREGION.')
      IF (ITC.EQ.0)  WRITE (6,16)
      IF (ITC.EQ.1)  WRITE (6,17)
   16 FORMAT (5X,'CURRENTS ARE CONSTANT IN EACH INDICATED SUBREGION.')
   17 FORMAT (5X,'CURRENTS ARE LINEAR IN EACH INDICATED SUBREGION.')
      IF (ITW.EQ.0)  WRITE (6,116)
      IF (ITW.EQ.1)  WRITE (6,117)
  116 FORMAT (/,5X,'WEIGHTING FLUX = FLUX;',/,5X,'WEIGHTING CURRENT = -
     XCURRENT.')
  117 FORMAT (/,5X,'WEIGHTING FLUX = ADJOINT FLUX;',/,5X,'WEIGHTING CURR
     XENT = ADJOINT CURRENT.')
```

```
      WRITE (6,20) EPS1,EPS2,EPS3,IPLOT,JPLOT,IPUNCH
   20 FORMAT (//,'0TOLERANCES TO POWER ARE :  EPS1 = ',1PD10.3,/,
     X    28X,'EPS2 = ',1PD10.3,/,28X,'EPS3 = ',1PD10.3,/,
     X    '0OUTPUT PARAMETERS TO POWER ARE:  IPLOT = ',I1,/,
     X    34X,'JPLOT = ',I1,/,34X,'IPUNCH = ',I1)
      WRITE (6,22) CHI(1), CHI(2),  THETA
   22 FORMAT (/,'0FISSION YIELDS ARE:  CHI(1) =',F10.5,/,
     X    22X,'CHI(2) =',F10.5,/,
     X    '0INPUT THETA PARAMETER (FOR MATRICES) =',D15.7)
      IF ((KR.LE.2).AND.(IBC.EQ.1))  CALL ERROR(1,KR)
      IF (KR.GT.25)  CALL ERROR(2,KR)
      IF (EPS1.LT.1.0E-16)  CALL  ERROR(6,1)
      IF (EPS2.LT.1.0E-16)  CALL  ERROR(6,2)
      IF (EPS3.LT.1.0E-16)  CALL  ERROR(6,3)
      IF ((IBC.LT.1).OR.(IBC.GT.4))  CALL ERROR(7,IBC)
C         DUMMY NORMAL VECTOR XU = UNITY.  (FOR THE INTEGRATION FUNCTIONS)
      DO 21  IG=1,2
      DO 21  II=1,100
   21 XU(IG,II)=1.0
      ITC0=2
      ITC1=2
      IF (ITC.EQ.1)  GO TO 23
      ITC0=0
      ITC1=1
C         COUNTER OF THE NUMBER OF TYPE-NUMBERS OF EACH TF REGION:
   23 NUMITF=1
      HX=0.0
C         BEGIN TO READ IN THE TF REGION DATA AND FILL THE ARRAYS,
C         DEPENDING ON THE TYPE-NUMBER OF EACH TF REGION.
      DO 50  I=1,KR
      IF (ITF(I).EQ.NUMITF)  GO TO 110
C         FILL THE ARRAYS FROM OLD TF REGION TYPES:
      J=ITF(I)
      CALL REPEAT(I,KTF(J))
      GO TO 50
C         READ IN THE TF REGION'S DATA FOR NEW TF REGION TYPE-NUMBERS:
```

```
  110 NUMITF=NUMITF+1
      KTF(NUMITF-1)=I
C        READ THE SUBREGION NUMBER AND THE NUMBER OF REGIONS IN THE SUBREGION:
      READ (5,1) K, KS
      IF (KS.GT.100)  CALL ERROR(3,I)
      KS1=KS+1
C        CHECK FOR IMPROPER SEQUENCING OF INPUT DATA:
      IF (I.NE.K)  CALL ERROR(4,I)
C        READ IN THE GEOMETRY AND THE MATERIAL PROPERTIES:
      READ (5,3) (X(J),X(J+1),H(J),A(1,J),F(1,J),D(1,J),S(1,J),
     X                              A(2,J),F(2,J),D(2,J),J=1,KS)
    3 FORMAT (3F10.5,4D10.3,/,30X,3D10.3)
C        READ IN THE REGIONAL GROUP TRIAL FUNCTIONS:
      READ (5,4) (PHI(1,J),CUR(1,J),PHI7(1,J),CUR7(1,J),J=1,KS1)
      READ (5,4) (PHI(2,J),CUR(2,J),PHI7(2,J),CUR7(2,J),J=1,KS1)
    4 FORMAT (4D20.7)
      IF (ITW.EQ.1)  GO TO 120
C        FORM WEIGHTING FUNCTIONS FROM THE GIVEN FUNCTIONS:
      DO 119  IG=1,2
      DO 119  J=1,KS1
      PHI7(IG,J)=PHI(IG,J)
  119 CUR7(IG,J)=-CUR(IG,J)
  120 IF (ITC.EQ.1)  GO TO 5
C        FORM THE REGION CONSTANT CURRENTS FROM THE FLUXES:
      DO 7  IG=1,2
      DO 6  J=1,KS
      CUR(IG,J)=-D(IG,J)*(-PHI(IG,J)+PHI(IG,J+1))/H(J)
    6 CUR7(IG,J)=+D(IG,J)*(-PHI7(IG,J)+PHI7(IG,J+1))/H(J)
      CUR(IG,KS1)=0.0
    7 CUR7(IG,KS1)=0.0
C        WRITE OUT THE INPUT INFORMATION IF ISEE .GE.2:
    5 IF (ISEE.LE.1)  GO TO 14
      WRITE (6,10)  K,KR,KS,(J,X(J),X(J+1),H(J),A(1,J),F(1,J),D(1,J),
     X   S(1,J),A(2,J),F(2,J),D(2,J),J=1,KS)
   10 FORMAT ('1INPUT MATERIAL PROPERTIES FOR REGION NUMBER ',I3,
     X   ', OF THE ',I3,' USED.',//,
```

```
     X   5X,'THIS REGION IS DIVIDED INTO ',I3,' HOMOGENEOUS SUBREGIONS A
     XS FOLLOWS:',//,
     X   5X,'FAST GROUP CONSTANTS APPEAR FIRST:',//,
     X   ' SUBREGION #',5X,'INTERNAL BOUNDARIES',10X,'WIDTH',3X,
     X   '  TOTAL CX (1/CM)',3X,'FISSION CX (1/CM)',6X,'DIFFUSION (CM)',
     X   4X,'SCATT.CX (1/CM)',/,
     X   5X,'I',11X,'X(I)',9X,'X(I+1)',11X,'H(I)',13X,'A(IG,I)',13X,
     X   'F(IG,I)',13X,'D(IG,I)',14X,'S(1,I)',//,
     X   (I6,3F15.4,4D20.8,/,51X,3D20.8))
      DO 15  IG=1,2
   15 WRITE (6,11) IG,K,KR,(J,X(J),PHI(IG,J),CUR(IG,J),PHI7(IG,J),
     X   CUR7(IG,J),J=1,KS1)
   11 FORMAT ('1INPUT TRIAL FUNCTIONS FOR GROUP',I2,' FOR REGION',I3,
     X   ' OUT OF THE',I3,' USED:',//,
     X   ' INDEX',5X,'COORD',16X,'FLUX',13X,'CURRENT',8X,' WEIGHT FLUX',
     X   5X,' WEIGHT CURRENT',//,(I6,F10.5,4D20.7))
   14 CONTINUE
C         END OF THE IN-OUT SECTION:
C         DEFINING MISC. ARRAYS FOR THE INTEGRATION FUNCTIONS:
C         LEGNTH OF THE SUBREGION: HT
      HT=X(KS1)-X(1)
      HR(K)=HT
      HX=HX+HR(K)
C         INVERSE OF THE D ARRAYS:
      DO 13  J=1,KS
      DI(1,J)=1./D(1,J)
   13 DI(2,J)=1./D(2,J)
C         FORMATION OF THE INTEGRATION FUNCTIONS:
      CALL BHSET(KS)
C         DO FOR ALL ENERGY GROUPS:
      DO 50  IG=1,2
      KA0(IG,K)=GIF(0,IG,PHI7,IG,A,PHI,KS,2)
      KA1(IG,K)=GIF(1,IG,PHI7,IG,A,PHI,KS,2)
      KA2(IG,K)=GIF(2,IG,PHI7,IG,A,PHI,KS,2)
      KA3(IG,K)=GIF(3,IG,PHI7,IG,A,PHI,KS,2)
      KA4(IG,K)=GIF(4,IG,PHI7,IG,A,PHI,KS,2)
```

```
      KA5(IG,K)=GIF(5,IG,PHI7,IG,A,PHI,KS,2)                      SYNT0181
      KA6(IG,K)=GIF(6,IG,PHI7,IG,A,PHI,KS,2)                      SYNT0182
      KB0(IG,K)=GIF(0,IG,PHI7,IG,F,PHI,KS,2)                      SYNT0183
      KB1(IG,K)=GIF(1,IG,PHI7,IG,F,PHI,KS,2)                      SYNT0184
      KB2(IG,K)=GIF(2,IG,PHI7,IG,F,PHI,KS,2)                      SYNT0185
      KB3(IG,K)=GIF(3,IG,PHI7,IG,F,PHI,KS,2)                      SYNT0186
      KB4(IG,K)=GIF(4,IG,PHI7,IG,F,PHI,KS,2)                      SYNT0187
      KB5(IG,K)=GIF(5,IG,PHI7,IG,F,PHI,KS,2)                      SYNT0188
      KB6(IG,K)=GIF(6,IG,PHI7,IG,F,PHI,KS,2)                      SYNT0189
      LA0(IG,K)=GIF(0,IG,CUR7,IG,DI,CUR,KS,ITC0)                  SYNT0190
      LA1(IG,K)=GIF(1,IG,CUR7,IG,DI,CUR,KS,ITC0)                  SYNT0191
      LA2(IG,K)=GIF(2,IG,CUR7,IG,DI,CUR,KS,ITC0)                  SYNT0192
      LA3(IG,K)=GIF(3,IG,CUR7,IG,DI,CUR,KS,ITC0)                  SYNT0193
      LA4(IG,K)=GIF(4,IG,CUR7,IG,DI,CUR,KS,ITC0)                  SYNT0194
      LA5(IG,K)=GIF(5,IG,CUR7,IG,DI,CUR,KS,ITC0)                  SYNT0195
      LA6(IG,K)=GIF(6,IG,CUR7,IG,DI,CUR,KS,ITC0)                  SYNT0196
       P0(IG,K)=GIF(0,IG,PHI7,IG,XU,CUR,KS,ITC1)/HT               SYNT0197
       P1(IG,K)=GIF(1,IG,PHI7,IG,XU,CUR,KS,ITC1)/HT               SYNT0198
       P2(IG,K)=GIF(2,IG,PHI7,IG,XU,CUR,KS,ITC1)/HT               SYNT0199
       P3(IG,K)=GIF(3,IG,PHI7,IG,XU,CUR,KS,ITC1)/HT               SYNT0200
       P4(IG,K)=GIF(4,IG,PHI7,IG,XU,CUR,KS,ITC1)/HT               SYNT0201
       P5(IG,K)=GIF(5,IG,PHI7,IG,XU,CUR,KS,ITC1)/HT               SYNT0202
       P6(IG,K)=GIF(6,IG,PHI7,IG,XU,CUR,KS,ITC1)/HT               SYNT0203
       Q0(IG,K)=GIF(0,IG,PHI,IG,XU,CUR7,KS,ITC1)/HT               SYNT0204
       Q1(IG,K)=GIF(1,IG,PHI,IG,XU,CUR7,KS,ITC1)/HT               SYNT0205
       Q2(IG,K)=GIF(2,IG,PHI,IG,XU,CUR7,KS,ITC1)/HT               SYNT0206
       Q3(IG,K)=GIF(3,IG,PHI,IG,XU,CUR7,KS,ITC1)/HT               SYNT0207
       Q4(IG,K)=GIF(4,IG,PHI,IG,XU,CUR7,KS,ITC1)/HT               SYNT0208
       Q5(IG,K)=GIF(5,IG,PHI,IG,XU,CUR7,KS,ITC1)/HT               SYNT0209
       Q6(IG,K)=GIF(6,IG,PHI,IG,XU,CUR7,KS,ITC1)/HT               SYNT0210
       R0(IG,K)=GIF(0,IG,PHI7,IG,D,PHI,KS,2)/HT**2                SYNT0211
       R1(IG,K)=GIF(1,IG,PHI7,IG,D,PHI,KS,2)/HT**2                SYNT0212
       R2(IG,K)=GIF(2,IG,PHI7,IG,D,PHI,KS,2)/HT**2                SYNT0213
       R3(IG,K)=GIF(3,IG,PHI7,IG,D,PHI,KS,2)/HT**2                SYNT0214
       R4(IG,K)=GIF(4,IG,PHI7,IG,D,PHI,KS,2)/HT**2                SYNT0215
C        STORE THE TERMINAL POINTS FOR LATER USE:                 SYNT0216
```

```
      PO(IG,K)=PHI(IG,1)
      PO7(IG,K)=PHI7(IG,1)
      PH(IG,K)=PHI(IG,KS1)
      PH7(IG,K)=PHI7(IG,KS1)
      DO(IG,K)=D(IG,1)
      DH(IG,K)=D(IG,KS)
      IF (K.EQ.1)  CO(IG)=CUR(IG,1)
      IF (NUMITF-1.EQ.ITF(KR).AND.ITC.EQ.0)  CH(IG)=CUR(IG,KS)
      IF (NUMITF-1.EQ.ITF(KR).AND.ITC.EQ.1)  CH(IG)=CUR(IG,KS1)
C        FOR THE OFF DIAGONAL MATRIX ELEMENTS:
      IF (IG.EQ.2)  GO TO 50
      SR0( 1,K)=GIF(0,2,PHI7,1,S,PHI,KS,2)
      SR1( 1,K)=GIF(1,2,PHI7,1,S,PHI,KS,2)
      SR2( 1,K)=GIF(2,2,PHI7,1,S,PHI,KS,2)
      SR3( 1,K)=GIF(3,2,PHI7,1,S,PHI,KS,2)
      SR4( 1,K)=GIF(4,2,PHI7,1,S,PHI,KS,2)
      SR5( 1,K)=GIF(5,2,PHI7,1,S,PHI,KS,2)
      SR6( 1,K)=GIF(6,2,PHI7,1,S,PHI,KS,2)
      KC0( 1,K)=GIF(0,1,PHI7,2,F,PHI,KS,2)
      KC1( 1,K)=GIF(1,1,PHI7,2,F,PHI,KS,2)
      KC2( 1,K)=GIF(2,1,PHI7,2,F,PHI,KS,2)
      KC3( 1,K)=GIF(3,1,PHI7,2,F,PHI,KS,2)
      KC4( 1,K)=GIF(4,1,PHI7,2,F,PHI,KS,2)
      KC5( 1,K)=GIF(5,1,PHI7,2,F,PHI,KS,2)
      KC6( 1,K)=GIF(6,1,PHI7,2,F,PHI,KS,2)
      KD0( 1,K)=GIF(0,2,PHI7,1,F,PHI,KS,2)
      KD1( 1,K)=GIF(1,2,PHI7,1,F,PHI,KS,2)
      KD2( 1,K)=GIF(2,2,PHI7,1,F,PHI,KS,2)
      KD3( 1,K)=GIF(3,2,PHI7,1,F,PHI,KS,2)
      KD4( 1,K)=GIF(4,2,PHI7,1,F,PHI,KS,2)
      KD5( 1,K)=GIF(5,2,PHI7,1,F,PHI,KS,2)
      KD6( 1,K)=GIF(6,2,PHI7,1,F,PHI,KS,2)
   50 CONTINUE
      NUMITF=NUMITF-1
      WRITE (6,51) NUMITF
   51 FORMAT ('1THERE ARE ONLY',I3,' DIFFERENT TRIAL FUNCTION REGIONS.')
```

```
      WRITE (6,52) (I,ITF(I),I=1,KR)                                    SYNT0253
   52 FORMAT (/,'OTABLE OF THE TRIAL FUNCTION NUMBER TYPES:',//,         SYNT0254
     X   3X,'TF REGION',4X,'REGION TYPE-NUMBER',//,                      SYNT0255
     X   (I7,12X,I7))                                                    SYNT0256
C        TO PRINT OUT THE /B5/ ARRAYS:                                   SYNT0257
      IF (ISEE.GE.2)  CALL PRTOUT(1)                                     SYNT0258
C        DETERMINATIONS OF THE B.C. OPTION PARAMETERS:                   SYNT0259
C        NN IS THE BLOCK SIZE OF THE POWER MATRICES.                     SYNT0260
      NN=2*KR                                                            SYNT0261
C        FORMATION OF THE COEFFICIENT VECTORS:                           SYNT0262
C        FILLING THE MATRICES FOR POWER:                                 SYNT0263
C        FOR BOTH ENERGY GROUPS:                                         SYNT0264
      DO 60  IG=1,2                                                      SYNT0265
C        THE MATRIX ROW INDEX:                                           SYNT0266
      I=1                                                                SYNT0267
C        FOR ALL THE INTERIOR COEFFICIENTS:                              SYNT0268
      DO 60  K=2,KR                                                      SYNT0269
      I=I+1                                                              SYNT0270
   56 J=K-1                                                              SYNT0271
      V=1./(PH7(IG,J)*PO(IG,J))                                          SYNT0272
      V1=1./(PH7(IG,J)*PH(IG,J))                                         SYNT0273
      V2=1./(PO7(IG,K)*PO(IG,K))                                         SYNT0274
      V3=1./(PO7(IG,K)*PH(IG,K))                                         SYNT0275
      LT(I,1,IG)=(3.*KA2(IG,J)-2.*KA3(IG,J)-9.*KA4(IG,J)+12.*KA5(IG,J)  SYNT0276
     X   -4.*KA6(IG,J)-(3.*LA2(IG,J)-2.*LA3(IG,J)-9.*LA4(IG,J)+12.*     SYNT0277
     X   LA5(IG,J)-4.*LA6(IG,J))-(6.*P1(IG,J)-6.*P2(IG,J)-18.*P3(IG,J)  SYNT0278
     X   +30.*P4(IG,J)-12.*P5(IG,J))-(18.*Q3(IG,J)-30.*Q4(IG,J)+12.*    SYNT0279
     X   Q5(IG,J))-(36.*R2(IG,J)-72.*R3(IG,J)+36.*R4(IG,J)))*V          SYNT0280
      LT(I,2,IG)=(-3.*KA3(IG,J)+8.*KA4(IG,J)-7.*KA5(IG,J)+2.*KA6(IG,J)  SYNT0281
     X   -(-3.*LA3(IG,J)+8.*LA4(IG,J)-7.*LA5(IG,J)+2.*LA6(IG,J))        SYNT0282
     X   -(-6.*P2(IG,J)+18.*P3(IG,J)-18.*P4(IG,J)+6.*P5(IG,J))          SYNT0283
     X   -(3.*Q2(IG,J)-14.*Q3(IG,J)+17.*Q4(IG,J)-6.*Q5(IG,J))           SYNT0284
     X   -(6.*R1(IG,J)-30.*                                             SYNT0285
     X   R2(IG,J)+42.*R3(IG,J)-18.*R4(IG,J)))*V*HR(J)/DO(IG,J)          SYNT0286
      LT(I,3,IG)=(9.*KA4(IG,J)-12.*KA5(IG,J)+4.*KA6(IG,J)-(9.*LA4(IG,J) SYNT0287
     X   -12.*LA5(IG,J)+4.*LA6(IG,J))-(18.*P3(IG,J)-30.*P4(IG,J)+12.*   SYNT0288
```

```
X    P5(IG,J))+(18.*Q3(IG,J)-30.*Q4(IG,J)+12.*Q5(IG,J))
X    +36.*R2(IG,J)-72.*R3(IG,J)+36.*R4(IG,J))*V1 +
X    (KA0(IG,K)-6.*KA2(IG,K)+4.*KA3(IG,K)+9.*KA4(IG,K)-12.*KA5(IG,K)
X    +4.*KA6(IG,K)-(LA0(IG,K)-6.*LA2(IG,K)+4.*LA3(IG,K)+9.*LA4(IG,K)
X    -12.*LA5(IG,K)+4.*LA6(IG,K))+6.*P1(IG,K)-6.*P2(IG,K)-18.*
X    P3(IG,K)+30.*P4(IG,K)-12.*P5(IG,K)-(6.*Q1(IG,K)-6.*Q2(IG,K)
X    -18.*Q3(IG,K)+30.*Q4(IG,K)-12.*Q5(IG,K))+36.*R2(IG,K)-72.*
X    R3(IG,K)+36.*R4(IG,K))*V2
 LT(I,4,IG)=(3.*KA4(IG,J)-5.*KA5(IG,J)+2.*KA6(IG,J)-(3.*LA4(IG,J)
X    -5.*LA5(IG,J)+2.*LA6(IG,J))-(6.*P3(IG,J)-12.*P4(IG,J)+6.*
X    P5(IG,J))-(-6.*Q3(IG,J)+13.*Q4(IG,J)-6.*Q5(IG,J))-(
X    -12.*R2(IG,J)
X    +30.*R3(IG,J)-18.*R4(IG,J)))*V1*HR(J)/DH(IG,J)
X    +(-KA1(IG,K)+2.*KA2(IG,K)+2.*KA3(IG,K)-8.*KA4(IG,K)+7.*
X    KA5(IG,K)-2.*KA6(IG,K)-(-LA1(IG,K)+2.*LA2(IG,K)+2.*LA3(IG,K)
X    -8.*LA4(IG,K)+7.*LA5(IG,K)-2.*LA6(IG,K))-6.*P2(IG,K)+18.*
X    P3(IG,K)-18.*P4(IG,K)+6.*P5(IG,K)-(Q0(IG,K)-4.*Q1(IG,K)+14.*
X    Q3(IG,K)-17.*Q4(IG,K)+6.*Q5(IG,K))+6.*R1(IG,K)-30.*R2(IG,K)
X    +42.*R3(IG,K)-18.*R4(IG,K))*V2*HR(K)/D0(IG,K)
 LT(I,5,IG)=(3.*KA2(IG,K)-2.*KA3(IG,K)-9.*KA4(IG,K)+12.*KA5(IG,K)
X    -4.*KA6(IG,K)-(3.*LA2(IG,K)-2.*LA3(IG,K)-9.*LA4(IG,K)+12.*
X    LA5(IG,K)-4.*LA6(IG,K))+18.*P3(IG,K)-30.*P4(IG,K)+12.*P5(IG,K)
X    +(6.*Q1(IG,K)-6.*Q2(IG,K)-18.*Q3(IG,K)+30.*Q4(IG,K)-12.*
X    Q5(IG,K))-(36.*R2(IG,K)-72.*R3(IG,K)+36.*R4(IG,K)))*V3
 LT(I,6,IG)=(KA2(IG,K)-KA3(IG,K)-3.*KA4(IG,K)+5.*KA5(IG,K)-2.*
X    KA6(IG,K)-(LA2(IG,K)-LA3(IG,K)-3.*LA4(IG,K)+5.*LA5(IG,K)-2.*
X    LA6(IG,K))+6.*P3(IG,K)-12.*P4(IG,K)+6.*P5(IG,K)-(-2.*Q1(IG,K)+
X    3.*Q2(IG,K)+6.*Q3(IG,K)-13.*Q4(IG,K)+6.*Q5(IG,K))-12.*R2(IG,K)
X    +30.*R3(IG,K)-18.*R4(IG,K))*V3*HR(K)/DH(IG,K)
 FT(I,1,IG)=(3.*KB2(IG,J)-2.*KB3(IG,J)-9.*KB4(IG,J)+12.*KB5(IG,J)
X    -4.*KB6(IG,J))*V
 FT(I,2,IG)=(-3.*KB3(IG,J)+8.*KB4(IG,J)-7.*KB5(IG,J)+2.*KB6(IG,J))
X    *V*HR(J)/D0(IG,J)
 FT(I,3,IG)=(9.*KB4(IG,J)-12.*KB5(IG,J)+4.*KB6(IG,J))*V1
X    +(KB0(IG,K)-6.*KB2(IG,K)+4.*KB3(IG,K)+9.*KB4(IG,K)
X    -12.*KB5(IG,K)+4.*KB6(IG,K))*V2
```

```
      FT(I,4,IG)=(3.*KB4(IG,J)-5.*KB5(IG,J)+2.*KB6(IG,J))               SYNT0325
     X    *V1*HR(J)/DH(IG,J)+(-KB1(IG,K)+2.*KB2(IG,K)+2.*KB3(IG,K)        SYNT0326
     X    -8.*KB4(IG,K)+7.*KB5(IG,K)-2.*KB6(IG,K))*V2*HR(K)/DO(IG,K)      SYNT0327
      FT(I,5,IG)=(3.*KB2(IG,K)-2.*KB3(IG,K)-9.*KB4(IG,K)+12.*KB5(IG,K)  SYNT0328
     X    -4.*KB6(IG,K))*V3                                              SYNT0329
      FT(I,6,IG)=(KB2(IG,K)-KB3(IG,K)-3.*KB4(IG,K)+5.*KB5(IG,K)         SYNT0330
     X    -2.*KB6(IG,K))*V3*HR(K)/DH(IG,K)                               SYNT0331
      IF (IG.EQ.2)  GO TO 57                                             SYNT0332
      T(I,1)    =(3.*SR2(IG,J)-2.*SR3(IG,J)-9.*SR4(IG,J)+12.*SR5(IG,J)  SYNT0333
     X    -4.*SR6(IG,J))/(PH7(2,J)*PO(1,J))                              SYNT0334
      T(I,2)    =(-3.*SR3(IG,J)+8.*SR4(IG,J)-7.*SR5(IG,J)+2.*SR6(IG,J)) SYNT0335
     X    *HR(J)/(PH7(2,J)*PO(1,J)*DO(1,J))                              SYNT0336
      T(I,3)    =(9.*SR4(IG,J)-12.*SR5(IG,J)+4.*SR6(IG,J))               SYNT0337
     X    /(PH7(2,J)*PH(1,J))                                            SYNT0338
     X    +(SR0(IG,K)-6.*SR2(IG,K)+4.*SR3(IG,K)+9.*SR4(IG,K)             SYNT0339
     X    -12.*SR5(IG,K)+4.*SR6(IG,K))/(PO7(2,K)*PO(1,K))               SYNT0340
      T(I,4)    =(3.*SR4(IG,J)-5.*SR5(IG,J)+2.*SR6(IG,J))                SYNT0341
     X    *HR(J)/(PH7(2,J)*PH(1,J)*DH(1,J))                              SYNT0342
     X    +(-SR1(IG,K)+2.*SR2(IG,K)+2.*SR3(IG,K)                         SYNT0343
     X    -8.*SR4(IG,K)+7.*SR5(IG,K)-2.*SR6(IG,K))*HR(K)                 SYNT0344
     X    /(PO7(2,K)*PO(1,K)*DO(1,K))                                    SYNT0345
      T(I,5)    =(3.*SR2(IG,K)-2.*SR3(IG,K)-9.*SR4(IG,K)+12.*SR5(IG,K)  SYNT0346
     X    -4.*SR6(IG,K))/(PO7(2,K)*PH(1,K))                              SYNT0347
      T(I,6)    =(SR2(IG,K)-SR3(IG,K)-3.*SR4(IG,K)+5.*SR5(IG,K)         SYNT0348
     X    -2.*SR6(IG,K))*HR(K)/(PO7(2,K)*PH(1,K)*DH(1,K))               SYNT0349
      FT(I,1, 3)=(3.*KD2(IG,J)-2.*KD3(IG,J)-9.*KD4(IG,J)+12.*KD5(IG,J)  SYNT0350
     X    -4.*KD6(IG,J))/(PH7(2,J)*PO(1,J))                              SYNT0351
      FT(I,2, 3)=(-3.*KD3(IG,J)+8.*KD4(IG,J)-7.*KD5(IG,J)+2.*KD6(IG,J)) SYNT0352
     X    *HR(J)/(PH7(2,J)*PO(1,J)*DO(1,J))                              SYNT0353
      FT(I,3, 3)=(9.*KD4(IG,J)-12.*KD5(IG,J)+4.*KD6(IG,J))               SYNT0354
     X    /(PH7(2,J)*PH(1,J))                                            SYNT0355
     X    +(KD0(IG,K)-6.*KD2(IG,K)+4.*KD3(IG,K)+9.*KD4(IG,K)             SYNT0356
     X    -12.*KD5(IG,K)+4.*KD6(IG,K))/(PO7(2,K)*PO(1,K))               SYNT0357
      FT(I,4, 3)=(3.*KD4(IG,J)-5.*KD5(IG,J)+2.*KD6(IG,J))                SYNT0358
     X    *HR(J)/(PH7(2,J)*PH(1,J)*DH(1,J))                              SYNT0359
     X    +(-KD1(IG,K)+2.*KD2(IG,K)+2.*KD3(IG,K)                         SYNT0360
```

```
     X   -8.*KD4(IG,K)+7.*KD5(IG,K)-2.*KD6(IG,K))*HR(K)
     X   /(PO7(2,K)*PO(1,K)*DO(1,K))
      FT(I,5, 3)=(3.*KD2(IG,K)-2.*KD3(IG,K)-9.*KD4(IG,K)+12.*KD5(IG,K)
     X   -4.*KD6(IG,K))/(PO7(2,K)*PH(1,K))
      FT(I,6, 3)=(KD2(IG,K)-KD3(IG,K)-3.*KD4(IG,K)+5.*KD5(IG,K)
     X   -2.*KD6(IG,K))*HR(K)/(PO7(2,K)*PH(1,K)*DH(1,K))
      FT(I,1, 4)=(3.*KC2(IG,J)-2.*KC3(IG,J)-9.*KC4(IG,J)+12.*KC5(IG,J)
     X   -4.*KC6(IG,J))/(PH7(1,J)*PO(2,J))
      FT(I,2, 4)=(-3.*KC3(IG,J)+8.*KC4(IG,J)-7.*KC5(IG,J)+2.*KC6(IG,J))
     X   *HR(J)/(PH7(1,J)*PO(2,J)*DO(2,J))
      FT(I,3, 4)=(9.*KC4(IG,J)-12.*KC5(IG,J)+4.*KC6(IG,J))
     X   /(PH7(1,J)*PH(2,J))
     X   +(KCO(IG,K)-6.*KC2(IG,K)+4.*KC3(IG,K)+9.*KC4(IG,K)
     X   -12.*KC5(IG,K)+4.*KC6(IG,K))/(PO7(1,K)*PO(2,K))
      FT(I,4, 4)=(3.*KC4(IG,J)-5.*KC5(IG,J)+2.*KC6(IG,J))
     X   *HR(J)/(PH7(1,J)*PH(2,J)*DH(2,J))
     X   +(-KC1(IG,K)+2.*KC2(IG,K)+2.*KC3(IG,K)
     X   -8.*KC4(IG,K)+7.*KC5(IG,K)-2.*KC6(IG,K))*HR(K)
     X   /(PO7(1,K)*PO(2,K)*DO(2,K))
      FT(I,5, 4)=(3.*KC2(IG,K)-2.*KC3(IG,K)-9.*KC4(IG,K)+12.*KC5(IG,K)
     X   -4.*KC6(IG,K))/(PO7(1,K)*PH(2,K))
      FT(I,6, 4)=(KC2(IG,K)-KC3(IG,K)-3.*KC4(IG,K)+5.*KC5(IG,K)
     X   -2.*KC6(IG,K))*HR(K)/(PO7(1,K)*PH(2,K)*DH(2,K))
   57 I=I+1
      LT(I,1,IG)=(KA2(IG,J)-KA3(IG,J)-3.*KA4(IG,J)+5.*KA5(IG,J)-2.*
     X   KA6(IG,J)-(LA2(IG,J)-LA3(IG,J)-3.*LA4(IG,J)+5.*LA5(IG,J)-2.*
     X   LA6(IG,J))-2.*P1(IG,J)+3.*P2(IG,J)+6.*P3(IG,J)-13.*P4(IG,J)+6.*
     X   P5(IG,J)-(6.*Q3(IG,J)-12.*Q4(IG,J)+6.*Q5(IG,J))-12.*R2(IG,J)
     X   +30.*R3(IG,J)-18.*R4(IG,J))*V*HR(J)/DH(IG,J)
      LT(I,2,IG)=(-KA3(IG,J)+3.*KA4(IG,J)-3.*KA5(IG,J)+KA6(IG,J)
     X   -(-LA3(IG,J)+3.*LA4(IG,J)-3.*LA5(IG,J)+LA6(IG,J))+2.*P2(IG,J)
     X   -7.*P3(IG,J)+8.*P4(IG,J)-3.*P5(IG,J)-(Q2(IG,J)-5.*Q3(IG,J)
     X   +7.*Q4(IG,J)-3.*Q5(IG,J))-2.*R1(IG,J)+11.*R2(IG,J)-18.*
     X   R3(IG,J)+9.*R4(IG,J))*V*HR(J)*HR(J)/(DO(IG,J)*DH(IG,J))
      LT(I,3,IG)=(3.*KA4(IG,J)-5.*KA5(IG,J)+2.*KA6(IG,J)-(3.*LA4(IG,J)
     X   -5.*LA5(IG,J)+2.*LA6(IG,J))-6.*P3(IG,J)+13.*P4(IG,J)-6.*
```

```
X    P5(IG,J)+(6.*Q3(IG,J)-12.*Q4(IG,J)+6.*Q5(IG,J))-(-12.*R2(IG,J)       SYNT0397
X    +30.*R3(IG,J)-18.*R4(IG,J)))*V1*HR(J)/DH(IG,J)                       SYNT0398
X    +(-KA1(IG,K)+2.*KA2(IG,K)+2.*KA3(IG,K)-8.*KA4(IG,K)+7.*              SYNT0399
X    KA5(IG,K)-2.*KA6(IG,K)-(-LA1(IG,K)+2.*LA2(IG,K)+2.*LA3(IG,K)-8.      SYNT0400
X    *LA4(IG,K)+7.*LA5(IG,K)-2.*LA6(IG,K))+P0(IG,K)-4.*P1(IG,K)+14.*      SYNT0401
X    P3(IG,K)-17.*P4(IG,K)+6.*P5(IG,K)-(-6.*Q2(IG,K)+18.*Q3(IG,K)         SYNT0402
X    -18.*Q4(IG,K)+6.*Q5(IG,K))+6.*R1(IG,K)-30.*R2(IG,K)+42.*             SYNT0403
X    R3(IG,K)-18.*R4(IG,K))*V2*HR(K)/DO(IG,K)                             SYNT0404
  LT(I,4,IG)=(KA4(IG,J)-2.*KA5(IG,J)+KA6(IG,J)-(LA4(IG,J)-2.*             SYNT0405
X    LA5(IG,J)+LA6(IG,J))-2.*P3(IG,J)+5.*P4(IG,J)-3.*P5(IG,J)-(-2.*       SYNT0406
X    Q3(IG,J)+5.*Q4(IG,J)-3.*Q5(IG,J))+4.*R2(IG,J)-12.*R3(IG,J)+9.*       SYNT0407
X    R4(IG,J))*V1*(HR(J)/DH(IG,J))**2                                     SYNT0408
X    +(KA2(IG,K)-4.*KA3(IG,K)+6.*KA4(IG,K)-4.*KA5(IG,K)+KA6(IG,K)         SYNT0409
X    -(LA2(IG,K)-4.*LA3(IG,K)+6.*LA4(IG,K)-4.*LA5(IG,K)+LA6(IG,K))        SYNT0410
X    -P1(IG,K)+6.*P2(IG,K)-12.*P3(IG,K)+10.*P4(IG,K)-3.*P5(IG,K)          SYNT0411
X    -(-Q1(IG,K)+6.*Q2(IG,K)-12.*Q3(IG,K)+10.*Q4(IG,K)-3.*Q5(IG,K))       SYNT0412
X    +R0(IG,K)-8.*R1(IG,K)+22.*R2(IG,K)-24.*R3(IG,K)+9.*R4(IG,K))*        SYNT0413
X    V2*(HR(K)/DO(IG,K))**2                                               SYNT0414
  LT(I,5,IG)=(-3.*KA3(IG,K)+8.*KA4(IG,K)-7.*KA5(IG,K)+2.*KA6(IG,K)        SYNT0415
X    -(-3.*LA3(IG,K)+8.*LA4(IG,K)-7.*LA5(IG,K)+2.*LA6(IG,K))              SYNT0416
X    +3.*P2(IG,K)-14.*P3(IG,K)+17.*P4(IG,K)-6.*P5(IG,K)+(-6.*             SYNT0417
X    Q2(IG,K)+18.*Q3(IG,K)-18.*Q4(IG,K)+6.*Q5(IG,K))-(6.*R1(IG,K)         SYNT0418
X    -30.*R2(IG,K)+42.*R3(IG,K)-18.*R4(IG,K)))*V3*HR(K)/DO(IG,K)          SYNT0419
  LT(I,6,IG)=(-KA3(IG,K)+3.*KA4(IG,K)-3.*KA5(IG,K)+KA6(IG,K)              SYNT0420
X    -(-LA3(IG,K)+3.*LA4(IG,K)-3.*LA5(IG,K)+LA6(IG,K))                    SYNT0421
X    +P2(IG,K)-5.*P3(IG,K)+7.*P4(IG,K)-3.*P5(IG,K)-(2.*Q2(IG,K)           SYNT0422
X    -7.*Q3(IG,K)+8.*Q4(IG,K)-3.*Q5(IG,K))-2.*R1(IG,K)+11.*R2(IG,K)       SYNT0423
X    -18.*R3(IG,K)+9.*R4(IG,K))*V3*HR(K)*HR(K)/(DO(IG,K)*DH(IG,K))        SYNT0424
  FT(I,1,IG)=(KB2(IG,J)-KB3(IG,J)-3.*KB4(IG,J)+5.*KB5(IG,J)               SYNT0425
X    -2.*KB6(IG,J))*V*HR(J)/DH(IG,J)                                      SYNT0426
  FT(I,2,IG)=(-KB3(IG,J)+3.*KB4(IG,J)-3.*KB5(IG,J)+KB6(IG,J))             SYNT0427
X    *V*HR(J)*HR(J)/(DO(IG,J)*DH(IG,J))                                   SYNT0428
  FT(I,3,IG)=(3.*KB4(IG,J)-5.*KB5(IG,J)+2.*KB6(IG,J))                     SYNT0429
X    *V1*HR(J)/DH(IG,J)+(-KB1(IG,K)+2.*KB2(IG,K)+2.*KB3(IG,K)             SYNT0430
X    -8.*KB4(IG,K)+7.*KB5(IG,K)-2.*KB6(IG,K))*V2*HR(K)/DO(IG,K)           SYNT0431
  FT(I,4,IG)=(KB4(IG,J)-2.*KB5(IG,J)+KB6(IG,J))                           SYNT0432
```

```
X    *V1*(HR(J)/DH(IG,J))**2+(KB2(IG,K)-4.*KB3(IG,K)+6.*KB4(IG,K)       SYNT0433
X    -4.*KB5(IG,K)+KB6(IG,K))*V2*(HR(K)/DO(IG,K))**2                    SYNT0434
 FT(I,5,IG)=(-3.*KB3(IG,K)+8.*KB4(IG,K)-7.*KB5(IG,K)+2.*KB6(IG,K))      SYNT0435
X    *V3*HR(K)/DO(IG,K)                                                 SYNT0436
 FT(I,6,IG)=(-KB3(IG,K)+3.*KB4(IG,K)-3.*KB5(IG,K)+KB6(IG,K))            SYNT0437
X    *V3*HR(K)*HR(K)/(DO(IG,K)*DH(IG,K))                                SYNT0438
 IF (IG.EQ.2)  GO TO 60                                                 SYNT0439
 T(I,1)=(SR2(IG,J)-SR3(IG,J)-3.*SR4(IG,J)+5.*SR5(IG,J)                  SYNT0440
X    -2.*SR6(IG,J))*HR(J)/(PH7(2,J)*PC(1,J)*DH(2,J))                    SYNT0441
 T(I,2)=(-SR3(IG,J)+3.*SR4(IG,J)-3.*SR5(IG,J)+SR6(IG,J))                SYNT0442
X    *HR(J)**2/(PH7(2,J)*PO(1,J)*DO(1,J)*DH(2,J))                       SYNT0443
 T(I,3)=(3.*SR4(IG,J)-5.*SR5(IG,J)+2.*SR6(IG,J))                        SYNT0444
X    *HR(J)/(PH7(2,J)*PH(1,J)*DH(2,J))                                  SYNT0445
X    +(-SR1(IG,K)+2.*SR2(IG,K)+2.*SR3(IG,K)                             SYNT0446
X    -8.*SR4(IG,K)+7.*SR5(IG,K)-2.*SR6(IG,K))*HR(K)                     SYNT0447
X    /(PO7(2,K)*PO(1,K)*DO(2,K))                                        SYNT0448
 T(I,4)=(SR4(IG,J)-2.*SR5(IG,J)+SR6(IG,J))                              SYNT0449
X    *HR(J)**2/(PH7(2,J)*PH(1,J)*DH(2,J)*DH(1,J))                       SYNT0450
X    +(SR2(IG,K)-4.*SR3(IG,K)                                           SYNT0451
X    +6.*SR4(IG,K)-4.*SR5(IG,K)+SR6(IG,K))                              SYNT0452
X    *HR(K)**2/(PO7(2,K)*PO(1,K)*DO(2,K)*DO(1,K))                       SYNT0453
 T(I,5)=(-3.*SR3(IG,K)+8.*SR4(IG,K)-7.*SR5(IG,K)+2.*SR6(IG,K))          SYNT0454
X    *HR(K)/(PO7(2,K)*PH(1,K)*DO(2,K))                                  SYNT0455
 T(I,6)=(-SR3(IG,K)+3.*SR4(IG,K)-3.*SR5(IG,K)+SR6(IG,K))                SYNT0456
X    *HR(K)**2/(PO7(2,K)*PH(1,K)*DO(2,K)*DH(1,K))                       SYNT0457
 FT(I,1, 3)=(KD2(IG,J)-KD3(IG,J)-3.*KD4(IG,J)+5.*KD5(IG,J)              SYNT0458
X    -2.*KD6(IG,J))*HR(J)/(PH7(2,J)*PC(1,J)*DH(2,J))                    SYNT0459
 FT(I,2, 3)=(-KD3(IG,J)+3.*KD4(IG,J)-3.*KD5(IG,J)+KD6(IG,J))            SYNT0460
X    *HR(J)**2/(PH7(2,J)*PO(1,J)*DO(1,J)*DH(2,J))                       SYNT0461
 FT(I,3, 3)=(3.*KD4(IG,J)-5.*KD5(IG,J)+2.*KD6(IG,J))                    SYNT0462
X    *HR(J)/(PH7(2,J)*PH(1,J)*DH(2,J))                                  SYNT0463
X    +(-KD1(IG,K)+2.*KD2(IG,K)+2.*KD3(IG,K)                             SYNT0464
X    -8.*KD4(IG,K)+7.*KD5(IG,K)-2.*KD6(IG,K))*HR(K)                     SYNT0465
X    /(PO7(2,K)*PO(1,K)*DO(2,K))                                        SYNT0466
 FT(I,4, 3)=(KD4(IG,J)-2.*KD5(IG,J)+KD6(IG,J))                          SYNT0467
X    *HR(J)**2/(PH7(2,J)*PH(1,J)*DH(2,J)*DH(1,J))                       SYNT0468
```

```
X     +(KD2(IG,K)-4.*KD3(IG,K)+6.*KD4(IG,K)                           SYNT0469
X     -4.*KD5(IG,K)+KD6(IG,K))*HR(K)**2                               SYNT0470
X     /(PO7(2,K)*PO(1,K)*DO(2,K)*DO(1,K))                             SYNT0471
      FT(I,5, 3)=(-3.*KD3(IG,K)+8.*KD4(IG,K)-7.*KD5(IG,K)+2.*KD6(IG,K)) SYNT0472
X     *HR(K)/(PO7(2,K)*PH(1,K)*DO(2,K))                               SYNT0473
      FT(I,6, 3)=(-KD3(IG,K)+3.*KD4(IG,K)-3.*KD5(IG,K)+KD6(IG,K))     SYNT0474
X     *HR(K)**2/(PO7(2,K)*PH(1,K)*DO(2,K)*DH(1,K))                    SYNT0475
      FT(I,1, 4)=(KC2(IG,J)-KC3(IG,J)-3.*KC4(IG,J)+5.*KC5(IG,J)       SYNT0476
X     -2.*KC6(IG,J))*HR(J)/(PH7(1,J)*PC(2,J)*DH(1,J))                 SYNT0477
      FT(I,2, 4)=(-KC3(IG,J)+3.*KC4(IG,J)-3.*KC5(IG,J)+KC6(IG,J))     SYNT0478
X     *HR(J)**2/(PH7(1,J)*PO(2,J)*DO(2,J)*DH(1,J))                    SYNT0479
      FT(I,3, 4)=(3.*KC4(IG,J)-5.*KC5(IG,J)+2.*KC6(IG,J))             SYNT0480
X     *HR(J)/(PH7(1,J)*PH(2,J)*DH(1,J))                               SYNT0481
X     +(-KC1(IG,K)+2.*KC2(IG,K)+2.*KC3(IG,K)                          SYNT0482
X     -8.*KC4(IG,K)+7.*KC5(IG,K)-2.*KC6(IG,K))*HR(K)                  SYNT0483
X     /(PO7(1,K)*PO(2,K)*DO(1,K))                                     SYNT0484
      FT(I,4, 4)=(KC4(IG,J)-2.*KC5(IG,J)+KC6(IG,J))                   SYNT0485
X     *HR(J)**2/(PH7(1,J)*PH(2,J)*DH(1,J)*DH(2,J))                    SYNT0486
X     +(KC2(IG,K)-4.*KC3(IG,K)+6.*KC4(IG,K)                           SYNT0487
X     -4.*KC5(IG,K)+KC6(IG,K))*HR(K)**2                               SYNT0488
X     /(PO7(1,K)*PO(2,K)*DO(1,K)*DO(2,K))                             SYNT0489
      FT(I,5, 4)=(-3.*KC3(IG,K)+8.*KC4(IG,K)-7.*KC5(IG,K)+2.*KC6(IG,K)) SYNT0490
X     *HR(K)/(PO7(1,K)*PH(2,K)*DO(1,K))                               SYNT0491
      FT(I,6, 4)=(-KC3(IG,K)+3.*KC4(IG,K)-3.*KC5(IG,K)+KC6(IG,K))     SYNT0492
X     *HR(K)**2/(PO7(1,K)*PH(2,K)*DO(1,K)*DH(2,K))                    SYNT0493
   60 CONTINUE                                                        SYNT0494
      IF (IBC.EQ.4)  GO TO 63                                         SYNT0495
C        ZERO FLUX COEFFICIENTS ON THE LEFT:                          SYNT0496
      DO 61  IG=1,2                                                   SYNT0497
      V2=1./(PO7(IG,1)*PO(IG,1))                                      SYNT0498
      V3=1./(PO7(IG,1)*PH(IG,1))                                      SYNT0499
      LT(1,4,IG)=(KA2(IG,1)-4.*KA3(IG,1)+6.*KA4(IG,1)-4.*KA5(IG,1)    SYNT0500
X     +KA6(IG,1)-(LA2(IG,1)-4.*LA3(IG,1)+6.*LA4(IG,1)-4.*LA5(IG,1)    SYNT0501
X     +LA6(IG,1))-P1(IG,1)+6.*P2(IG,1)-12.*P3(IG,1)+10.*P4(IG,1)      SYNT0502
X     -3.*P5(IG,1)-(-Q1(IG,1)+6.*Q2(IG,1)-12.*Q3(IG,1)+10.*Q4(IG,1)   SYNT0503
X     -3.*Q5(IG,1))+R0(IG,1)-8.*R1(IG,1)+22.*R2(IG,1)-24.*R3(IG,1)    SYNT0504
```

```
X     +9.*R4(IG,1))*V2*(HR(1)/DO(IG,1))**2                                    SYNT0505
      LT(1,5,IG)=(-3.*KA3(IG,1)+8.*KA4(IG,1)-7.*KA5(IG,1)+2.*KA6(IG,1)        SYNT0506
X     -(-3.*LA3(IG,1)+8.*LA4(IG,1)-7.*LA5(IG,1)+2.*LA6(IG,1))                 SYNT0507
X     +3.*P2(IG,1)-14.*P3(IG,1)+17.*P4(IG,1)-6.*P5(IG,1)+(-6.*                SYNT0508
X     Q2(IG,1)+18.*Q3(IG,1)-18.*Q4(IG,1)+6.*Q5(IG,1))-(6.*R1(IG,1)            SYNT0509
X     -30.*R2(IG,1)+42.*R3(IG,1)-18.*R4(IG,1)))*V3*HR(1)/DO(IG,1)             SYNT0510
      LT(1,6,IG)=(-KA3(IG,1)+3.*KA4(IG,1)-3.*KA5(IG,1)+KA6(IG,1)              SYNT0511
X     -(-LA3(IG,1)+3.*LA4(IG,1)-3.*LA5(IG,1)+LA6(IG,1))+P2(IG,1)              SYNT0512
X     -5.*P3(IG,1)+7.*P4(IG,1)-3.*P5(IG,1)-(2.*Q2(IG,1)-7.*Q3(IG,1)           SYNT0513
X     +8.*Q4(IG,1)-3.*Q5(IG,1))-2.*R1(IG,1)+11.*R2(IG,1)-18.*R3(IG,1)         SYNT0514
X     +9.*R4(IG,1))*V3*HR(1)*HR(1)/(DO(IG,1)*DH(IG,1))                        SYNT0515
      FT(1,4,IG)=(KB2(IG,1)-4.*KB3(IG,1)+6.*KB4(IG,1)-4.*KB5(IG,1)            SYNT0516
X     +KB6(IG,1))*V2*(HR(1)/DO(IG,1))**2                                      SYNT0517
      FT(1,5,IG)=(-3.*KB3(IG,1)+8.*KB4(IG,1)-7.*KB5(IG,1)+2.*KB6(IG,1))       SYNT0518
X     *V3*HR(1)/DO(IG,1)                                                      SYNT0519
      FT(1,6,IG)=(-KB3(IG,1)+3.*KB4(IG,1)-3.*KB5(IG,1)+KB6(IG,1))             SYNT0520
X     *V3*HR(1)*HR(1)/(DO(IG,1)*DH(IG,1))                                     SYNT0521
      IF (IG.EQ.2)  GO TO 61                                                  SYNT0522
      T(1,4)=(SR2(IG,1)-4.*SR3(IG,1)+6.*SR4(IG,1)-4.*SR5(IG,1)                SYNT0523
X     +SR6(IG,1))*HR(1)**2/(PO7(2,1)*PO(1,1)*DO(2,1)*DO(1,1))                 SYNT0524
      T(1,5)=(-3.*SR3(IG,1)+8.*SR4(IG,1)-7.*SR5(IG,1)+2.*SR6(IG,1))           SYNT0525
X     *HR(1)/(PO7(2,1)*PH(1,1)*DO(2,1))                                       SYNT0526
      T(1,6)=(-SR3(IG,1)+3.*SR4(IG,1)-3.*SR5(IG,1)+SR6(IG,1))                 SYNT0527
X     *HR(1)**2/(PO7(2,1)*PH(1,1)*DO(2,1)*DH(1,1))                            SYNT0528
      FT(1,4, 3)=(KD2(IG,1)-4.*KD3(IG,1)+6.*KD4(IG,1)-4.*KD5(IG,1)            SYNT0529
X     +KD6(IG,1))*HR(1)**2/(PO7(2,1)*PO(1,1)*DO(2,1)*DO(1,1))                 SYNT0530
      FT(1,5, 3)=(-3.*KD3(IG,1)+8.*KD4(IG,1)-7.*KD5(IG,1)+2.*KD6(IG,1))       SYNT0531
X     *HR(1)/(PO7(2,1)*PH(1,1)*DO(2,1))                                       SYNT0532
      FT(1,6, 3)=(-KD3(IG,1)+3.*KD4(IG,1)-3.*KD5(IG,1)+KD6(IG,1))             SYNT0533
X     *HR(1)**2/(PO7(2,1)*PH(1,1)*DO(2,1)*DH(1,1))                            SYNT0534
      FT(1,4, 4)=(KC2(IG,1)-4.*KC3(IG,1)+6.*KC4(IG,1)-4.*KC5(IG,1)            SYNT0535
X     +KC6(IG,1))*HR(1)**2/(PO7(1,1)*PO(2,1)*DO(1,1)*DO(2,1))                 SYNT0536
      FT(1,5, 4)=(-3.*KC3(IG,1)+8.*KC4(IG,1)-7.*KC5(IG,1)+2.*KC6(IG,1))       SYNT0537
X     *HR(1)/(PO7(1,1)*PH(2,1)*DO(1,1))                                       SYNT0538
      FT(1,6, 4)=(-KC3(IG,1)+3.*KC4(IG,1)-3.*KC5(IG,1)+KC6(IG,1))             SYNT0539
X     *HR(1)**2/(PO7(1,1)*PH(2,1)*DO(1,1)*DH(2,1))                            SYNT0540
```

```
   61 CONTINUE
      GO TO 65
C        ZERO CURRENT COEFFICIENTS ON THE LEFT:
   63 K=1
      I=1
      DO 64  IG=1,2
      V2=1./(PO7(IG,K)*PO(IG,K))
      V3=1./(PO7(IG,K)*PH(IG,K))
      LT(I,4,IG)=
     X   (KA0(IG,K)-6.*KA2(IG,K)+4.*KA3(IG,K)+9.*KA4(IG,K)-12.*KA5(IG,K)
     X   +4.*KA6(IG,K)-(LA0(IG,K)-6.*LA2(IG,K)+4.*LA3(IG,K)+9.*LA4(IG,K)
     X   -12.*LA5(IG,K)+4.*LA6(IG,K))+6.*P1(IG,K)-6.*P2(IG,K)-18.*
     X   P3(IG,K)+30.*P4(IG,K)-12.*P5(IG,K)-(6.*Q1(IG,K)-6.*Q2(IG,K)
     X   -18.*Q3(IG,K)+30.*Q4(IG,K)-12.*Q5(IG,K))+36.*R2(IG,K)-72.*
     X   R3(IG,K)+36.*R4(IG,K))*V2 -CO(IG)/PO(IG,1)
      LT(I,5,IG)=(3.*KA2(IG,K)-2.*KA3(IG,K)-9.*KA4(IG,K)+12.*KA5(IG,K)
     X   -4.*KA6(IG,K)-(3.*LA2(IG,K)-2.*LA3(IG,K)-9.*LA4(IG,K)+12.*
     X   LA5(IG,K)-4.*LA6(IG,K))+18.*P3(IG,K)-30.*P4(IG,K)+12.*P5(IG,K)
     X   +(6.*Q1(IG,K)-6.*Q2(IG,K)-18.*Q3(IG,K)+30.*Q4(IG,K)-12.*
     X   Q5(IG,K))-(36.*R2(IG,K)-72.*R3(IG,K)+36.*R4(IG,K)))*V3
      LT(I,6,IG)=(KA2(IG,K)-KA3(IG,K)-3.*KA4(IG,K)+5.*KA5(IG,K)-2.*
     X   KA6(IG,K)-(LA2(IG,K)-LA3(IG,K)-3.*LA4(IG,K)+5.*LA5(IG,K)-2.*
     X   LA6(IG,K))+6.*P3(IG,K)-12.*P4(IG,K)+6.*P5(IG,K)-(-2.*Q1(IG,K)+
     X   3.*Q2(IG,K)+6.*Q3(IG,K)-13.*Q4(IG,K)+6.*Q5(IG,K))-12.*R2(IG,K)
     X   +30.*R3(IG,K)-18.*R4(IG,K))*V3*HR(K)/DH(IG,K)
      FT(I,4,IG)=
     X   +(KB0(IG,K)-6.*KB2(IG,K)+4.*KB3(IG,K)+9.*KB4(IG,K)
     X   -12.*KB5(IG,K)+4.*KB6(IG,K))*V2
      FT(I,5,IG)=(3.*KB2(IG,K)-2.*KB3(IG,K)-9.*KB4(IG,K)+12.*KB5(IG,K)
     X   -4.*KB6(IG,K))*V3
      FT(I,6,IG)=(KB2(IG,K)-KB3(IG,K)-3.*KB4(IG,K)+5.*KB5(IG,K)
     X   -2.*KB6(IG,K))*V3*HR(K)/DH(IG,K)
      IF (IG.EQ.2)  GO TO 64
      T(I,4)=
     X   +(SR0(IG,K)-6.*SR2(IG,K)+4.*SR3(IG,K)+9.*SR4(IG,K)
     X   -12.*SR5(IG,K)+4.*SR6(IG,K))/(PO7(2,K)*PO(1,K))
```

```
      T(I,5)    =(3.*SR2(IG,K)-2.*SR3(IG,K)-9.*SR4(IG,K)+12.*SR5(IG,K)          SYNT0577
     X   -4.*SR6(IG,K))/(PO7(2,K)*PH(1,K))                                      SYNT0578
      T(I,6)    =(SR2(IG,K)-SR3(IG,K)-3.*SR4(IG,K)+5.*SR5(IG,K)                 SYNT0579
     X   -2.*SR6(IG,K))*HR(K)/(PO7(2,K)*PH(1,K)*DH(1,K))                        SYNT0580
      FT(I,4, 3)=                                                               SYNT0581
     X   +(KD0(IG,K)-6.*KD2(IG,K)+4.*KD3(IG,K)+9.*KD4(IG,K)                     SYNT0582
     X   -12.*KD5(IG,K)+4.*KD6(IG,K))/(PO7(2,K)*PO(1,K))                        SYNT0583
      FT(I,5, 3)=(3.*KD2(IG,K)-2.*KD3(IG,K)-9.*KD4(IG,K)+12.*KD5(IG,K)          SYNT0584
     X   -4.*KD6(IG,K))/(PO7(2,K)*PH(1,K))                                      SYNT0585
      FT(I,6, 3)=(KD2(IG,K)-KD3(IG,K)-3.*KD4(IG,K)+5.*KD5(IG,K)                 SYNT0586
     X   -2.*KD6(IG,K))*HR(K)/(PO7(2,K)*PH(1,K)*DH(1,K))                        SYNT0587
      FT(I,4, 4)=                                                               SYNT0588
     X   +(KC0(IG,K)-6.*KC2(IG,K)+4.*KC3(IG,K)+9.*KC4(IG,K)                     SYNT0589
     X   -12.*KC5(IG,K)+4.*KC6(IG,K))/(PO7(1,K)*PO(2,K))                        SYNT0590
      FT(I,5, 4)=(3.*KC2(IG,K)-2.*KC3(IG,K)-9.*KC4(IG,K)+12.*KC5(IG,K)          SYNT0591
     X   -4.*KC6(IG,K))/(PO7(1,K)*PH(2,K))                                      SYNT0592
      FT(I,6, 4)=(KC2(IG,K)-KC3(IG,K)-3.*KC4(IG,K)+5.*KC5(IG,K)                 SYNT0593
     X   -2.*KC6(IG,K))*HR(K)/(PO7(1,K)*PH(2,K)*DH(2,K))                        SYNT0594
   64 CONTINUE                                                                  SYNT0595
C        FIX UP THE TWO FIRST COLUMN ENTRIES TO MATCH F(1) (NOT G(1)):          SYNT0596
      DO 70  IG=1,2                                                             SYNT0597
      DO 70  I=2,3                                                              SYNT0598
      LT(I,2,IG)=LT(I,1,IG)                                                     SYNT0599
      FT(I,2,IG)=FT(I,1,IG)                                                     SYNT0600
      IF (IG.EQ.2)  GO TO 70                                                    SYNT0601
      T(I,2)=T(I,1)                                                             SYNT0602
      FT(I,2, 3)=FT(I,1, 3)                                                     SYNT0603
      FT(I,2, 4)=FT(I,1, 4)                                                     SYNT0604
   70 CONTINUE                                                                  SYNT0605
   65 I=2*KR                                                                    SYNT0606
      J=KR                                                                      SYNT0607
      K=KR                                                                      SYNT0608
      IF (IBC.EQ.2.OR.IBC.EQ.4)   GO TO 74                                      SYNT0609
C        ZERO FLUX COEFFICIENTS ON THE RIGHT:                                   SYNT0610
      DO 72  IG=1,2                                                             SYNT0611
      V=1./(PH7(IG,J)*PO(IG,J))                                                 SYNT0612
```

```
      V1=1./(PH7(IG,J)*PH(IG,J))                                          SYNT0613
      LT(I,1,IG)=(KA2(IG,J)-KA3(IG,J)-3.*KA4(IG,J)+5.*KA5(IG,J)-2.*      SYNT0614
     X   KA6(IG,J)-(LA2(IG,J)-LA3(IG,J)-3.*LA4(IG,J)+5.*LA5(IG,J)-2.*    SYNT0615
     X   LA6(IG,J))-2.*P1(IG,J)+3.*P2(IG,J)+6.*P3(IG,J)-13.*P4(IG,J)+6.* SYNT0616
     X   P5(IG,J)-(6.*Q3(IG,J)-12.*Q4(IG,J)+6.*Q5(IG,J))-12.*R2(IG,J)    SYNT0617
     X   +30.*R3(IG,J)-18.*R4(IG,J))*V*HR(J)/DH(IG,J)                    SYNT0618
      LT(I,2,IG)=(-KA3(IG,J)+3.*KA4(IG,J)-3.*KA5(IG,J)+KA6(IG,J)         SYNT0619
     X   -(-LA3(IG,J)+3.*LA4(IG,J)-3.*LA5(IG,J)+LA6(IG,J))+2.*P2(IG,J)   SYNT0620
     X   -7.*P3(IG,J)+8.*P4(IG,J)-3.*P5(IG,J)-(Q2(IG,J)-5.*Q3(IG,J)      SYNT0621
     X   +7.*Q4(IG,J)-3.*Q5(IG,J) )-2.*R1(IG,J)+11.*R2(IG,J)-18.*        SYNT0622
     X   R3(IG,J)+9.*R4(IG,J))*V*HR(J)*HR(J)/(DO(IG,J)*DH(IG,J))         SYNT0623
      LT(I,3,IG)=(KA4(IG,J)-2.*KA5(IG,J)+KA6(IG,J)-(LA4(IG,J)-2.*        SYNT0624
     X   LA5(IG,J)+LA6(IG,J))-2.*P3(IG,J)+5.*P4(IG,J)-3.*P5(IG,J)-(-2.*  SYNT0625
     X   Q3(IG,J)+5.*Q4(IG,J)-3.*Q5(IG,J))+4.*R2(IG,J)-12.*R3(IG,J)+9.*  SYNT0626
     X   R4(IG,J))*V1*(HR(J)/DH(IG,J))**2                                SYNT0627
      FT(I,1,IG)=(KB2(IG,J)-KB3(IG,J)-3.*KB4(IG,J)+5.*KB5(IG,J)          SYNT0628
     X   -2.*KB6(IG,J))*V*HR(J)/DH(IG,J)                                 SYNT0629
      FT(I,2,IG)=(-KB3(IG,J)+3.*KB4(IG,J)-3.*KB5(IG,J)+KB6(IG,J))        SYNT0630
     X   *V*HR(J)*HR(J)/(DO(IG,J)*DH(IG,J))                              SYNT0631
      FT(I,3,IG)=(KB4(IG,J)-2.*KB5(IG,J)+KB6(IG,J))                      SYNT0632
     X   *V1*(HR(J)/DH(IG,J))**2                                         SYNT0633
      IF (IG.EQ.2)  GO TO 72                                             SYNT0634
      T(I,1)=(SR2(IG,J)-SR3(IG,J)-3.*SR4(IG,J)+5.*SR5(IG,J)              SYNT0635
     X   -2.*SR6(IG,J))*HR(J)/(PH7(2,J)*PC(1,J)*DH(2,J))                 SYNT0636
      T(I,2)=(-SR3(IG,J)+3.*SR4(IG,J)-3.*SR5(IG,J)+SR6(IG,J))            SYNT0637
     X   *HR(J)**2/(PH7(2,J)*PO(1,J)*DH(2,J)*DO(1,J))                    SYNT0638
      T(I,3)=(SR4(IG,J)-2.*SR5(IG,J)+SR6(IG,J))                          SYNT0639
     X   *HR(J)**2/(PH7(2,J)*PH(1,J)*DH(2,J)*DH(1,J))                    SYNT0640
      FT(I,1, 3)=(KD2(IG,J)-KD3(IG,J)-3.*KD4(IG,J)+5.*KD5(IG,J)          SYNT0641
     X   -2.*KD6(IG,J))*HR(J)/(PH7(2,J)*PC(1,J)*DH(2,J))                 SYNT0642
      FT(I,2, 3)=(-KD3(IG,J)+3.*KD4(IG,J)-3.*KD5(IG,J)+KD6(IG,J))        SYNT0643
     X   *HR(J)**2/(PH7(2,J)*PO(1,J)*DH(2,J)*DO(1,J))                    SYNT0644
      FT(I,3, 3)=(KD4(IG,J)-2.*KD5(IG,J)+KD6(IG,J))                      SYNT0645
     X   *HR(J)**2/(PH7(2,J)*PH(1,J)*DH(2,J)*DH(1,J))                    SYNT0646
      FT(I,1, 4)=(KC2(IG,J)-KC3(IG,J)-3.*KC4(IG,J)+5.*KC5(IG,J)          SYNT0647
     X   -2.*KC6(IG,J))*HR(J)/(PH7(1,J)*PC(2,J)*DH(1,J))                 SYNT0648
```

```
      FT(I,2, 4)=(-KC3(IG,J)+3.*KC4(IG,J)-3.*KC5(IG,J)+KC6(IG,J))            SYNT0649
     X   *HR(J)**2/(PH7(1,J)*PO(2,J)*DH(1,J)*DO(2,J))                        SYNT0650
      FT(I,3, 4)=(KC4(IG,J)-2.*KC5(IG,J)+KC6(IG,J))                          SYNT0651
     X   *HR(J)**2/(PH7(1,J)*PH(2,J)*DH(1,J)*DH(2,J))                        SYNT0652
   72 CONTINUE                                                               SYNT0653
C        FIX UP THE LAST TWO COLUMNS TO MATCH G(K+1) (NOT F(K+1)):           SYNT0654
      I1=2*KR-2                                                              SYNT0655
      I2=I1+1                                                                SYNT0656
      DO 73  IG=1,2                                                          SYNT0657
      DO 73  I=I1,I2                                                         SYNT0658
      LT(I,5,IG)=LT(I,6,IG)                                                  SYNT0659
      FT(I,5,IG)=FT(I,6,IG)                                                  SYNT0660
      IF (IG.EQ.2)  GO TO 73                                                 SYNT0661
      T(I,5)=T(I,6)                                                          SYNT0662
      FT(I,5, 3)=FT(I,6, 3)                                                  SYNT0663
      FT(I,5, 4)=FT(I,6, 4)                                                  SYNT0664
   73 CONTINUE                                                               SYNT0665
      GO TO 80                                                               SYNT0666
C        ZERO CURRENT COEFFICIENTS ON THE RIGHT:                             SYNT0667
   74 DO 62  IG=1,2                                                          SYNT0668
      V=1./(PH7(IG,K)*PO(IG,K))                                              SYNT0669
      V1=1./(PH7(IG,K)*PH(IG,K))                                             SYNT0670
      LT(I,1,IG)=(3.*KA2(IG,K)-2.*KA3(IG,K)-9.*KA4(IG,K)+12.*KA5(IG,K)       SYNT0671
     X   -4.*KA6(IG,K)-(3.*LA2(IG,K)-2.*LA3(IG,K)-9.*LA4(IG,K)+12.*          SYNT0672
     X   LA5(IG,K)-4.*LA6(IG,K))-(6.*P1(IG,K)-6.*P2(IG,K)-18.*P3(IG,K)       SYNT0673
     X   +30.*P4(IG,K)-12.*P5(IG,K))-(18.*Q3(IG,K)-30.*Q4(IG,K)+12.*         SYNT0674
     X   Q5(IG,K))-(36.*R2(IG,K)-72.*R3(IG,K)+36.*R4(IG,K)))*V               SYNT0675
      LT(I,2,IG)=(-3.*KA3(IG,K)+8.*KA4(IG,K)-7.*KA5(IG,K)+2.*KA6(IG,K)       SYNT0676
     X   -(-3.*LA3(IG,K)+8.*LA4(IG,K)-7.*LA5(IG,K)+2.*LA6(IG,K))-(-6.*       SYNT0677
     X   P2(IG,K)+18.*P3(IG,K)-18.*P4(IG,K)+6.*P5(IG,K))-(3.*Q2(IG,K)        SYNT0678
     X   -14.*Q3(IG,K)+17.*Q4(IG,K)-6.*Q5(IG,K))-(6.*R1(IG,K)-30.*           SYNT0679
     X   R2(IG,K)+42.*R3(IG,K)-18.*R4(IG,K)))*V*HR(K)/DO(IG,K)               SYNT0680
      LT(I,3,IG)=(9.*KA4(IG,K)-12.*KA5(IG,K)+4.*KA6(IG,K)-(9.*LA4(IG,K)      SYNT0681
     X   -12.*LA5(IG,K)+4.*LA6(IG,K))-(18.*P3(IG,K)-30.*P4(IG,K)+12.*        SYNT0682
     X   P5(IG,K))+(18.*Q3(IG,K)-30.*Q4(IG,K)+12.*Q5(IG,K))                  SYNT0683
     X   +36.*R2(IG,K)-72.*R3(IG,K)+36.*R4(IG,K))*V1  +CH(IG)/PH(IG,KR)      SYNT0684
```

```
      FT(I,1,IG)=(3.*KB2(IG,K)-2.*KB3(IG,K)-9.*KB4(IG,K)+12.*KB5(IG,K)
     X    -4.*KB6(IG,K))*V
      FT(I,2,IG)=(-3.*KB3(IG,K)+8.*KB4(IG,K)-7.*KB5(IG,K)+2.*KB6(IG,K))
     X    *V*HR(K)/DO(IG,K)
      FT(I,3,IG)=(9.*KB4(IG,K)-12.*KB5(IG,K)+4.*KB6(IG,K))*V1
      IF (IG.EQ.2)  GO TO 62
      T(I,1)=(3.*SR2(IG,K)-2.*SR3(IG,K)-9.*SR4(IG,K)+12.*SR5(IG,K)
     X    -4.*SR6(IG,K))/(PH7(2,K)*PO(1,K))
      T(I,2)=(-3.*SR3(IG,K)+8.*SR4(IG,K)-7.*SR5(IG,K)+2.*SR6(IG,K))
     X    *HR(K)/(PH7(2,K)*PO(1,K)*DO(1,K))
      T(I,3)=(9.*SR4(IG,K)-12.*SR5(IG,K)+4.*SR6(IG,K))
     X    /(PH7(2,K)*PH(1,K))
      FT(I,1, 3)=(3.*KD2(IG,K)-2.*KD3(IG,K)-9.*KD4(IG,K)+12.*KD5(IG,K)
     X    -4.*KD6(IG,K))/(PH7(2,K)*PO(1,K))
      FT(I,2, 3)=(-3.*KD3(IG,K)+8.*KD4(IG,K)-7.*KD5(IG,K)+2.*KD6(IG,K))
     X    *HR(K)/(PH7(2,K)*PO(1,K)*DO(1,K))
      FT(I,3, 3)=(9.*KD4(IG,K)-12.*KD5(IG,K)+4.*KD6(IG,K))
     X    /(PH7(2,K)*PH(1,K))
      FT(I,1, 4)=(3.*KC2(IG,K)-2.*KC3(IG,K)-9.*KC4(IG,K)+12.*KC5(IG,K)
     X    -4.*KC6(IG,K))/(PH7(1,K)*PO(2,K))
      FT(I,2, 4)=(-3.*KC3(IG,K)+8.*KC4(IG,K)-7.*KC5(IG,K)+2.*KC6(IG,K))
     X    *HR(K)/(PH7(1,K)*PO(2,K)*DO(2,K))
      FT(I,3, 4)=(9.*KC4(IG,K)-12.*KC5(IG,K)+4.*KC6(IG,K))
     X    /(PH7(1,K)*PH(2,K))
   62 CONTINUE
   80 CONTINUE
C        INCLUDE THETA AND PHI (PHI = -1) IN THE MATRIX FORMATIONS:
C        THE ABOVE EQUATIONS ARE DERIVED USING PHI = -1.
      PHIPHI=+1.0D0
      IF (THETA.NE.1.0)  CALL MATFIX(THETA,PHIPHI)
C        TO PRINT OUT THE /B3/ MATRICES:
      IF (ISEE.GE.2)  CALL PRTOUT(2)
      RETURN
      END
```

```
      SUBROUTINE ERROR(I,J)
C        ANNOUNCES INPUT ERRORS AND TERMINATES PROGRAM EXECUTION:
      GO TO (1,2,3,4,5,6,7,8,9),I
    1 WRITE (6,101)
      GO TO 10
    2 WRITE (6,102) J
      GO TO 10
    3 WRITE (6,103) J
      GO TO 10
    4 WRITE (6,104) J
      GO TO 10
    5 WRITE (6,105) J
      GO TO 10
    6 WRITE (6,106) J
      GO TO 10
    7 CONTINUE
    8 CONTINUE
    9 CONTINUE
   10 WRITE (6,110)
  101 FORMAT ('1MUST HAVE > 2 SUBREGIONS FOR ZERO FLUX B.C.S. INVALID.')
  102 FORMAT ('1NUMBER OF SUBREGIONS =',I3,' > 25. INVALID.')
  103 FORMAT ('1SUBREGION NUMBER',I3,' HAS > 100 SECTIONS. INVALID.')
  104 FORMAT ('1INPUT ERROR IN REGION SEQUENCING AT REGION',I5,'.')
  105 FORMAT ('1Z(I) = 0. IN REGION I =',I3,'. INVALID.')
  106 FORMAT ('1THE TOLERANCE:  EPS',I1,'  IS < 1.0E-16.  INVALID.')
  107 FORMAT ('1BOUNDRY CONDITION OPTION =',I2,' < 1 OR > 4. INVALID.')
  110 FORMAT (1H0,'PROBLEM TERMINATED.')
      CALL EXIT
      RETURN
      END
```

```
      SUBROUTINE REPEAT(K,L)                                            REPE0001
C     SETS THE /B5/ ARRAYS (K) EQUAL TO PAST STORED ARRAYS (L):         REPE0002
      IMPLICIT REAL*8 (A-Z)                                             REPE0003
      COMMON /B5/                                                       REPE0004
     X   KA0(2,25),KA1(2,25),KA2(2,25),KA3(2,25),KA4(2,25),KA5(2,25),   REPE0005
     X   KA6(2,25),KB0(2,25),KB1(2,25),KB2(2,25),KB3(2,25),KB4(2,25),   REPE0006
     X   KB5(2,25),KB6(2,25),LA0(2,25),LA1(2,25),LA2(2,25),LA3(2,25),   REPE0007
     X   LA4(2,25),LA5(2,25),LA6(2,25),P0(2,25) ,P1(2,25) ,P2(2,25) ,   REPE0008
     X   P3(2,25) ,P4(2,25) ,P5(2,25) ,P6(2,25) ,Q0(2,25) ,Q1(2,25) ,   REPE0009
     X   Q2(2,25) ,Q3(2,25) ,Q4(2,25) ,Q5(2,25) ,Q6(2,25) ,R0(2,25) ,   REPE0010
     X   R1(2,25) ,R2(2,25) ,R3(2,25) ,R4(2,25) ,SR0(1,25),SR1(1,25),   REPE0011
     X   SR2(1,25),SR3(1,25),SR4(1,25),SR5(1,25),SR6(1,25),KC0(1,25),   REPE0012
     X   KC1(1,25),KC2(1,25),KC3(1,25),KC4(1,25),KC5(1,25),KC6(1,25),   REPE0013
     X   KD0(1,25),KD1(1,25),KD2(1,25),KD3(1,25),KD4(1,25),KD5(1,25),   REPE0014
     X   KD6(1,25),                                                     REPE0015
     X   P0(2,25) ,PH(2,25) ,P07(2,25),PH7(2,25),D0(2,25) ,DH(2,25)     REPE0016
      COMMON /XAXIS/ HX,HR(25)                                          REPE0017
      INTEGER K,L,G                                                     REPE0018
      HR(K)=HR(L)                                                       REPE0019
      HX=HX+HR(K)                                                       REPE0020
      DO 10  G=1,2                                                      REPE0021
      KA0(G,K)=KA0(G,L)                                                 REPE0022
      KA1(G,K)=KA1(G,L)                                                 REPE0023
      KA2(G,K)=KA2(G,L)                                                 REPE0024
      KA3(G,K)=KA3(G,L)                                                 REPE0025
      KA4(G,K)=KA4(G,L)                                                 REPE0026
      KA5(G,K)=KA5(G,L)                                                 REPE0027
      KA6(G,K)=KA6(G,L)                                                 REPE0028
      KB0(G,K)=KB0(G,L)                                                 REPE0029
      KB1(G,K)=KB1(G,L)                                                 REPE0030
      KB2(G,K)=KB2(G,L)                                                 REPE0031
      KB3(G,K)=KB3(G,L)                                                 REPE0032
      KB4(G,K)=KB4(G,L)                                                 REPE0033
      KB5(G,K)=KB5(G,L)                                                 REPE0034
      KB6(G,K)=KB6(G,L)                                                 REPE0035
      LA0(G,K)=LA0(G,L)                                                 REPE0036
```

```
      LA1(G,K)=LA1(G,L)                                                 REPE0037
      LA2(G,K)=LA2(G,L)                                                 REPE0038
      LA3(G,K)=LA3(G,L)                                                 REPE0039
      LA4(G,K)=LA4(G,L)                                                 REPE0040
      LA5(G,K)=LA5(G,L)                                                 REPE0041
      LA6(G,K)=LA6(G,L)                                                 REPE0042
      P0(G,K)=P0(G,L)                                                   REPE0043
      P1(G,K)=P1(G,L)                                                   REPE0044
      P2(G,K)=P2(G,L)                                                   REPE0045
      P3(G,K)=P3(G,L)                                                   REPE0046
      P4(G,K)=P4(G,L)                                                   REPE0047
      P5(G,K)=P5(G,L)                                                   REPE0048
      P6(G,K)=P6(G,L)                                                   REPE0049
      Q0(G,K)=Q0(G,L)                                                   REPE0050
      Q1(G,K)=Q1(G,L)                                                   REPE0051
      Q2(G,K)=Q2(G,L)                                                   REPE0052
      Q3(G,K)=Q3(G,L)                                                   REPE0053
      Q4(G,K)=Q4(G,L)                                                   REPE0054
      Q5(G,K)=Q5(G,L)                                                   REPE0055
      Q6(G,K)=Q6(G,L)                                                   REPE0056
      R0(G,K)=R0(G,L)                                                   REPE0057
      R1(G,K)=R1(G,L)                                                   REPE0058
      R2(G,K)=R2(G,L)                                                   REPE0059
      R3(G,K)=R3(G,L)                                                   REPE0060
      R4(G,K)=R4(G,L)                                                   REPE0061
      IF (G.EQ.2)  GO TO 5                                              REPE0062
      SR0(G,K)=SR0(G,L)                                                 REPE0063
      SR1(G,K)=SR1(G,L)                                                 REPE0064
      SR2(G,K)=SR2(G,L)                                                 REPE0065
      SR3(G,K)=SR3(G,L)                                                 REPE0066
      SR4(G,K)=SR4(G,L)                                                 REPE0067
      SR5(G,K)=SR5(G,L)                                                 REPE0068
      SR6(G,K)=SR6(G,L)                                                 REPE0069
      KC0(G,K)=KC0(G,L)                                                 REPE0070
      KC1(G,K)=KC1(G,L)                                                 REPE0071
      KC2(G,K)=KC2(G,L)                                                 REPE0072
```

```
      KC3(G,K)=KC3(G,L)
      KC4(G,K)=KC4(G,L)
      KC5(G,K)=KC5(G,L)
      KC6(G,K)=KC6(G,L)
      KD0(G,K)=KD0(G,L)
      KD1(G,K)=KD1(G,L)
      KD2(G,K)=KD2(G,L)
      KD3(G,K)=KD3(G,L)
      KD4(G,K)=KD4(G,L)
      KD5(G,K)=KD5(G,L)
      KD6(G,K)=KD6(G,L)
    5 CONTINUE
      PO(G,K)=PO(G,L)
      PO7(G,K)=PO7(G,L)
      PH(G,K)=PH(G,L)
      PH7(G,K)=PH7(G,L)
      DO(G,K)=DO(G,L)
      DH(G,K)=DH(G,L)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BHSET(K)                                               BHSE0001
C        SETS UP THE /BH/ ARRAYS FOR GIF:                               BHSE0002
      IMPLICIT REAL*8 (A-H,L-Z)                                         BHSE0003
      COMMON /BH/ X(101), H(101), Z(101)                                BHSE0004
      DO 1  I=1,K                                                       BHSE0005
    1 Z(I)=X(I)-X(1)                                                    BHSE0006
      RETURN                                                            BHSE0007
      END                                                               BHSE0008
```

```
      DOUBLE PRECISION FUNCTION GIF(N,G1,F,G2,C,G,K,ITC)                  GIF 0001
C                                                                         GIF 0002
C     *  IDENTICAL TO SUBROUTINE GIF PREVIOUSLY LISTED IN PROGRAM LINEAR.  GIF 0003
C                                                                         GIF 0004
      RETURN                                                              GIF 0005
      END                                                                 GIF 0006
```

```
      DOUBLE PRECISION FUNCTION FACT(N)                                 FACT0001
C     COMPUTES N FACTORIAL:                                             FACT0002
      FACT=1.0D0                                                        FACT0003
      IF (N.LE.1)  RETURN                                               FACT0004
      DO 1  I=2,N                                                       FACT0005
    1 FACT=FACT*DFLOAT(I)                                               FACT0006
      RETURN                                                            FACT0007
      END                                                               FACT0008
```

```
      SUBROUTINE MATFIX(THEATA,PHI)
C     MODIFYS THE MATRIX ELEMENTS OF THE /B3/ MATRICES BY THEATA AND PHI.
C        PROPER CHOICE OF THEATA PROVIDES EASIER INVERSION OF THE MATRICES.
C     MATRIX SOLUTION SHOULD BE INDEPENDENT OF PHI.  HOWEVER:
C     USE OF PHI > 0 RESULTS IN POSITIVE DEFINITE MATRICES.
      COMMON /B1/ IBC
      COMMON /B2/ KR,NN
      COMMON /B3/ A(50,6,7)
      REAL*8 A,THEATA,PHI,X,Y,Z
      NO=NN-1
      DO 10  L=1,7
      DO 5  I=2,NO
      X=THEATA
      Y=1.0D0
      IF (MOD(I,2).EQ.0)  GO TO 1
      X=THEATA*PHI
      Y=PHI
    1 DO 2  M=2,6,2
    2 A(I,M,L)=X*A(I,M,L)
      DO 3  M=1,5,2
    3 A(I,M,L)=Y*A(I,M,L)
    5 CONTINUE
C        BOUNDARY CONDITION EQUATIONS:
      X=1.0D0
      Y=1.0D0
      Z=THEATA
      IF (IBC.GT.2)  GO TO 6
      X=THEATA*PHI
      Y=PHI
      Z=X
    6 A(1,4,L)=X*A(1,4,L)
      A(1,5,L)=Y*A(1,5,L)
      A(1,6,L)=Z*A(1,6,L)
      X=1.0D0
      Y=THEATA
      Z=X
```

```
      IF (IBC.EQ.2.OR.IBC.EQ.4)  GO TO 7
      X=PHI
      Y=THEATA*PHI
      Z=Y
    7 A(NN,1,L)=X*A(NN,1,L)
      A(NN,2,L)=Y*A(NN,2,L)
      A(NN,3,L)=Z*A(NN,3,L)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PRTOUT(IP)
C        IP = 1  PRINTS OUT THE /B5/ ARRAYS USED IN MATRIX FORMATIONS.
C        IP = 2  PRINTS OUT THE /B3/ MATRICES GIVEN TO POWER.
      IMPLICIT REAL*8 (A-H,K-Z)
      COMMON /B2/ KR, N
      COMMON /B3/ L1(50,6), L2(50,6), F1(50,6), F4(50,6), F3(50,6),
     X            F2(50,6), T(50,6)
      COMMON /B5/
     X   KA0(2,25),KA1(2,25),KA2(2,25),KA3(2,25),KA4(2,25),KA5(2,25),
     X   KA6(2,25),KB0(2,25),KB1(2,25),KB2(2,25),KB3(2,25),KB4(2,25),
     X   KB5(2,25),KB6(2,25),LA0(2,25),LA1(2,25),LA2(2,25),LA3(2,25),
     X   LA4(2,25),LA5(2,25),LA6(2,25),P0(2,25) ,P1(2,25) ,P2(2,25) ,
     X   P3(2,25) ,P4(2,25) ,P5(2,25) ,P6(2,25) ,Q0(2,25) ,Q1(2,25) ,
     X   Q2(2,25) ,Q3(2,25) ,Q4(2,25) ,Q5(2,25) ,Q6(2,25) ,R0(2,25) ,
     X   R1(2,25) ,R2(2,25) ,R3(2,25) ,R4(2,25) ,SR0(1,25),SR1(1,25),
     X   SR2(1,25),SR3(1,25),SR4(1,25),SR5(1,25),SR6(1,25),KC0(1,25),
     X   KC1(1,25),KC2(1,25),KC3(1,25),KC4(1,25),KC5(1,25),KC6(1,25),
     X   KD0(1,25),KD1(1,25),KD2(1,25),KD3(1,25),KD4(1,25),KD5(1,25),
     X   KD6(1,25),
     X   P07(2,25) ,PH(2,25) ,P07(2,25),PH7(2,25),D0(2,25) ,DH(2,25)
      COMMON /XAXIS/ HX,HR(25)
      INTEGER KR, G, N, K
      GO TO (1001,1002), IP
C        KA'S:
 1001 WRITE (6,10)
      DO 11  G=1,2
   11 WRITE (6,100) (G,K,KA0(G,K),KA1(G,K),KA2(G,K),KA3(G,K),KA4(G,K),
     X   KA5(G,K),KA6(G,K),K=1,KR)
C        KB'S:
      WRITE (6,20)
      DO 21  G=1,2
   21 WRITE (6,100) (G,K,KB0(G,K),KB1(G,K),KB2(G,K),KB3(G,K),KB4(G,K),
     X   KB5(G,K),KB6(G,K),K=1,KR)
C        KC'S:
      WRITE (6,22)
      G=1
```

```
   23 WRITE (6,100) (G,K,KC0(G,K),KC1(G,K),KC2(G,K),KC3(G,K),KC4(G,K),
     X   KC5(G,K),KC6(G,K),K=1,KR)
C          KD'S:
      WRITE (6,24)
      G=1
   25 WRITE (6,100) (G,K,KD0(G,K),KD1(G,K),KD2(G,K),KD3(G,K),KD4(G,K),
     X   KD5(G,K),KD6(G,K),K=1,KR)
C          LA'S:
      WRITE (6,30)
      DO 31  G=1,2
   31 WRITE (6,100) (G,K,LA0(G,K),LA1(G,K),LA2(G,K),LA3(G,K),LA4(G,K),
     X   LA5(G,K),LA6(G,K),K=1,KR)
C          SR'S:
      WRITE (6,40)
      G=1
   41 WRITE (6,100) (G,K,SR0(G,K),SR1(G,K),SR2(G,K),SR3(G,K),SR4(G,K),
     X   SR5(G,K),SR6(G,K),K=1,KR)
C          P'S:
      WRITE (6,50)
      DO 51  G=1,2
   51 WRITE (6,100) (G,K,P0(G,K),P1(G,K),P2(G,K),P3(G,K),P4(G,K),
     X   P5(G,K),P6(G,K),K=1,KR)
C          Q'S:
      WRITE (6,60)
      DO 61  G=1,2
   61 WRITE (6,100) (G,K,Q0(G,K),Q1(G,K),Q2(G,K),Q3(G,K),Q4(G,K),
     X   Q5(G,K),Q6(G,K),K=1,KR)
C          R'S:
      WRITE (6,70)
      DO 71  G=1,2
   71 WRITE (6,101) (G,K,R0(G,K),R1(G,K),R2(G,K),R3(G,K),R4(G,K),K=1,KR)
C          BOUNDARY VALUES:
      WRITE (6,80)
      DO 81  G=1,2
   81 WRITE (6,100) (G,K,P0(G,K),PH(G,K),P07(G,K),PH7(G,K),D0(G,K),
     X   DH(G,K),HR(K),K=1,KR)
```

```
      RETURN                                                            PRT00073
 1002 WRITE (6,90)                                                      PRT00074
      WRITE (6,110) ((L1(I,J),J=1,6),I=1,N)                             PRT00075
      WRITE (6,91)                                                      PRT00076
      WRITE (6,110) ((L2(I,J),J=1,6),I=1,N)                             PRT00077
      WRITE (6,92)                                                      PRT00078
      WRITE (6,110) ((F1(I,J),J=1,6),I=1,N)                             PRT00079
      WRITE (6,93)                                                      PRT00080
      WRITE (6,110) ((F2(I,J),J=1,6),I=1,N)                             PRT00081
      WRITE (6,94)                                                      PRT00082
      WRITE (6,110) ((F3(I,J),J=1,6),I=1,N)                             PRT00083
      WRITE (6,95)                                                      PRT00084
      WRITE (6,110) ((F4(I,J),J=1,6),I=1,N)                             PRT00085
      WRITE (6,96)                                                      PRT00086
      WRITE (6,110) (( T(I,J),J=1,6),I=1,N)                             PRT00087
   10 FORMAT ('1    G',4X,'K',7X,'KA0(G,K)',7X,'KA1(G,K)',7X,'KA2(G,K)',  PRT00088
     X    7X,'KA3(G,K)',7X,'KA4(G,K)',7X,'KA5(G,K)',7X,'KA6(G,K)',/)    PRT00089
   20 FORMAT ('1    G',4X,'K',7X,'KB0(G,K)',7X,'KB1(G,K)',7X,'KB2(G,K)',  PRT00090
     X    7X,'KB3(G,K)',7X,'KB4(G,K)',7X,'KB5(G,K)',7X,'KB6(G,K)',/)    PRT00091
   22 FORMAT ('1    G',4X,'K',7X,'KC0(G,K)',7X,'KC1(G,K)',7X,'KC2(G,K)',  PRT00092
     X    7X,'KC3(G,K)',7X,'KC4(G,K)',7X,'KC5(G,K)',7X,'KC6(G,K)',/)    PRT00093
   24 FORMAT ('1    G',4X,'K',7X,'KD0(G,K)',7X,'KD1(G,K)',7X,'KD2(G,K)',  PRT00094
     X    7X,'KD3(G,K)',7X,'KD4(G,K)',7X,'KD5(G,K)',7X,'KD6(G,K)',/)    PRT00095
   30 FORMAT ('1    G',4X,'K',7X,'LA0(G,K)',7X,'LA1(G,K)',7X,'LA2(G,K)',  PRT00096
     X    7X,'LA3(G,K)',7X,'LA4(G,K)',7X,'LA5(G,K)',7X,'LA6(G,K)',/)    PRT00097
   40 FORMAT ('1    G',4X,'K',7X,'SR0(G,K)',7X,'SR1(G,K)',7X,'SR2(G,K)',  PRT00098
     X    7X,'SR3(G,K)',7X,'SR4(G,K)',7X,'SR5(G,K)',7X,'SR6(G,K)',/)    PRT00099
   50 FORMAT ('1    G',4X,'K',8X,'P0(G,K)',8X,'P1(G,K)',8X,'P2(G,K)',8X,  PRT00100
     X    'P3(G,K)',8X,'P4(G,K)',8X,'P5(G,K)',8X,'P6(G,K)',/)           PRT00101
   60 FORMAT ('1    G',4X,'K',8X,'Q0(G,K)',8X,'Q1(G,K)',8X,'Q2(G,K)',8X,  PRT00102
     X    'Q3(G,K)',8X,'Q4(G,K)',8X,'Q5(G,K)',8X,'Q6(G,K)',/)           PRT00103
   70 FORMAT ('1    G',4X,'K',8X,'R0(G,K)',8X,'R1(G,K)',8X,'R2(G,K)',8X,  PRT00104
     X    'R3(G,K)',8X,'R4(G,K)',/)                                     PRT00105
   80 FORMAT ('1    G',4X,'K',8X,'P0(G,K)',8X,'PH(G,K)',7X,'P07(G,K)',7X, PRT00106
     X    'PH7(G,K)',8X,'D0(G,K)',8X,'DH(G,K)',10X,'HR(K)',/)           PRT00107
  100 FORMAT (2I5,7D15.7)                                               PRT00108
```

```
101 FORMAT (2I5,5D15.7)                                         PRT00109
 90 FORMAT ('1MATRIX L1:',/)                                    PRT00110
 91 FORMAT ('1MATRIX L2:',/)                                    PRT00111
 92 FORMAT ('1MATRIX F1:',/)                                    PRT00112
 93 FORMAT ('1MATRIX F2:',/)                                    PRT00113
 94 FORMAT ('1MATRIX F3:',/)                                    PRT00114
 95 FORMAT ('1MATRIX F4:',/)                                    PRT00115
 96 FORMAT ('1MATRIX  T:',/)                                    PRT00116
110 FORMAT (6D20.10)                                            PRT00117
    RETURN                                                      PRT00118
    END                                                         PRT00119
```

```
      SUBROUTINE POWER
C     SOLVES THE 2*N MULTIGROUP EQUATIONS:  M*PHI = (1/LAMDA)*F*PHI
C     BY THE FISSION SOURCE POWER METHOD
C     USING SIMULTANEOUS OVERRELAXATION.
C        WHERE:  M AND F ARE DOUBLE PRECISION 2N BY 2N BLOCK MATRICES;
C        AND:    PHI IS THE 2N FLUX (FAST AND THERMAL) VECTOR.
C                  L1*PHI1 = CHI1*(F1*PHI1 + F2*PHI2)
C        -T*PHI1 + L2*PHI2 = CHI2*(F3*PHI1 + F4*PHI2)
C        METHOD FOLLOWS WACHPRESS, PAGE 83.  SOLUTION BY GROUP ITERATION.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH,ISEE
      COMMON /B2/ KR,N
      COMMON /B3/ L1(50,6),L2(50,6),F1(50,6),F4(50,6),F3(50,6),F2(50,6),
     X            T(50,6)
      COMMON /B4/ PHI(2,52), PSI(2,52), LAMDA, ICOUT
      COMMON /B5/ S(52), ERROR(2,52), Z(52), G(50,6), GT(50)
      COMMON /B6/ TE1(2,5),TE2(2,5),TE3(5),IN(5)
      COMMON /CHIF/ CHI(2)
      COMMON /XAXIS/ HX,HR(25)
      COMMON /ER/ EPS1,EPS2,EPS3
      COMMON /FSTR/ PHISTR(2,26,6)
      COMMON /ESTR/ LAMSTR(300), EFSTR(2,300), EFMSTR(2,300), ERLAM(300)
      COMMON /TRUE/ TRULAM, TRUPHI(2,52), PHICON(2,300), LAMCON(300),IFT
      DIMENSION PSI1(52), PSI2(52), SQ(2), DPHI(2), ERRMAX(2)
      INTEGER N, NN
C        DEFAULT OPTIONS FOR THE TRUE EIGENVALUE AND FLUX-CURRENT VECTOR:
      TRULAM=1.0
      DO 5  IG=1,2
      DO 4  I=1,N
      TRUPHI(IG,I)=1.0
      IF (MOD(I,2).EQ.1)  TRUPHI(IG,I)=0.0
    4 CONTINUE
      IF (IBC.EQ.1)  TRUPHI(IG,N)=0.0
      IF (IBC.EQ.4)  TRUPHI(IG,1)=1.0
    5 CONTINUE
C        DEFAULT OPTIONS FOR POWER PARAMETERS:
```

```
      ALPHA=1.25
      LAMDA=1.0
      DO 555  IG=1,2
      IF (IBC.NE.4)  GO TO 551
      DO 550  I=1,N
      PHI(IG,I)=1.0
      IF (MOD(I,2).EQ.1)  PHI(IG,I)=0.0
  550 CONTINUE
      PHI(IG,1)=1.0
      GO TO 555
  551 X=3.1415926/HX
      IF (IBC.NE.1)  X=X/2.0
      PHI(IG,1)=-X
      SUM1=0.0
      DO 552  K=2,KR
      I=2*K-2
      J=I+1
      SUM1=SUM1+HR(K-1)
      PHI(IG,I)=DSIN(SUM1*X)
  552 PHI(IG,J)=-X*DCOS(SUM1*X)
      PHI(IG,N)=1.0
      IF (IBC.EQ.1)  PHI(IG,N)=X
  555 CONTINUE
C        READ IN THE TRUE (EXPECTED) EIGENVALUE AND FLUX VECTOR (MINUS 0 BC'S):
      IFT=0
      READ (5,500,END=501) TRULAM, (TRUPHI(1,I),I=1,N)
      READ (5,503,END=501) (TRUPHI(2,I),I=1,N)
      IFT=1
  500 FORMAT (E25.14,/,(4E20.10))
C        READ IN:  OVERRELAXATION PARAMETERS  ;  ALPHA  (OUTER ITERATION)
C                  INITIAL GUESS AT EIGENVALUE;  LAMDA
C                  INITIAL NORMALIZED FLUX    ;  PHI(1-N)
  501 READ (5,506,END=510) ALPHA
      READ (5,502,END=510) LAMDA
      READ (5,503) (PHI(1,I),I=1,N)
      READ (5,503) (PHI(2,I),I=1,N)
```

```
  506 FORMAT (F10.5)                                                         POWE0073
  502 FORMAT (E25.14)                                                        POWE0074
  503 FORMAT ((4E20.10))                                                     POWE0075
  510 CONTINUE                                                               POWE0076
C        STORING FOR PRINTING THE MULTIGROUP FLUX SHAPE.                     POWE0077
      K=0                                                                    POWE0078
      IF (IBC.EQ.4)  K=1                                                     POWE0079
      DO 11  IG=1,2                                                          POWE0080
      DO 10  I=1,KR                                                          POWE0081
      II=I+K                                                                 POWE0082
   10 PHISTR(IG,II,2)=PHI(IG,2*I)                                            POWE0083
      IF (IBC.EQ.4)  PHISTR(IG,1,2)=PHI(IG,1)                                POWE0084
C        FILL RUNNING COORD IN PHISTR                                        POWE0085
      KR1=KR+1                                                               POWE0086
      DO 11  I=1,KR1                                                         POWE0087
   11 PHISTR(IG,I,1)=DFLOAT(I)                                               POWE0088
C        IK IS THE FLUX PLOTTING COUNTER.                                    POWE0089
      IK=1                                                                   POWE0090
C        STORES THE ITERATION NUMBER FOR FLUX HISTORY PLOTTING:              POWE0091
      IN(1)=0                                                                POWE0092
C        STORES TEMPORARY ERRORS FOR FLUX HISTORY PLOTTING:                  POWE0093
      TE1(1,1)=0.                                                            POWE0094
      TE1(2,1)=0.                                                            POWE0095
      TE2(1,1)=0.                                                            POWE0096
      TE2(2,1)=0.                                                            POWE0097
      TE3(1)=0.0                                                             POWE0098
C        EIGENVALUE OF THE PREVIOUS ITERATION:                               POWE0099
      LAMB4=LAMDA                                                            POWE0100
C        THE MAXIMUM NUMBER OF ALLOWED ITERATIONS: ICMAX                     POWE0101
      ICMAX=300                                                              POWE0102
C        PRINT OUT THE POWER METHOD PARAMETER INFORMATION:                   POWE0103
      WRITE (6,700) ICMAX,ALPHA,LAMDA,(PHI(1,I),I=1,N)                       POWE0104
      WRITE (6,701) (PHI(2,I),I=1,N)                                         POWE0105
  700 FORMAT ('1EXECUTING MULTIGROUP FISSION SOURCE POWER ITERATION METH     POWE0106
     XOD.',///,                                                              POWE0107
     X   5X,'MAXIMUM NUMBER OF ALLOWABLE ITERATIONS:',/,                     POWE0108
```

```
     X   10X,'ICMAX =',I4,///,
     X   5X,'OUTER ITERATION RELAXATION PARAMETER:',/,
     X   10X,'ALPHA =',F7.3,//,
     X   5X,'INITIAL GUESS AT EIGENVALUE:',/,
     X   10X,'LAMBDA =',E22.14,//,
     X   5X,'INITIAL GUESS AT THE GROUP FLUX SHAPE CONNECTION POINTS:',
     X   //,8X,'FAST GROUP:',/,
     X   10X,'F(K)''S =',4E25.14,/,(18X,4E25.14))
  701 FORMAT ('0',7X,'THERMAL GROUP:',/,
     X   10X,'F(K)''S =',4E25.14,/,(18X,4E25.14))
C        BEGIN ITERATION LOOP.
      ICOUT=0
C        ICOUT IS THE OUTER ITERATION COUNTER.
   20 ICOUT=ICOUT+1
      IF (ICOUT.GT.ICMAX)  GO TO 100
C        SOLVE FOR THE NEW GROUP FLUX VECTORS: PSI:
C        THE GROUP FLUX ITERATES:
      DO 21  I=1,N
      PSI1(I)=PHI(1,I)
   21 PSI2(I)=PHI(2,I)
C        THE FAST ITERATION SOURCE S:
      CALL VPROD(F1,PSI1,S,N)
      CALL VPROD(F2,PSI2,Z,N)
      DO 22  I=1,N
   22 S(I)=CHI(1)*(S(I)+Z(I))
C        FAST FLUX:
      CALL SOLVE(L1,PSI1,S,N,G,GT)
C        THE THERMAL ITERATION SOURCE S:
      CALL VPROD(F3,PSI1,S,N)
      CALL VPROD(F4,PSI2,Z,N)
      DO 25  I=1,N
   25 S(I)=CHI(2)*(S(I)+Z(I))
      CALL VPROD(T,PSI1,Z,N)
      DO 27  I=1,N
   27 S(I)=S(I)+Z(I)
C        THERMAL FLUX:
```

```
      CALL SOLVE(L2,PSI2,S,N,G,GT)                                        POWE0145
C        CALCULATION OF THE EIGENVALUE:                                   POWE0146
      SUM1=0.0D0                                                          POWE0147
      SUM2=0.0D0                                                          POWE0148
      DO 28  I=1,N                                                        POWE0149
      SUM1=SUM1+PSI1(I)*PHI(1,I)                                          POWE0150
   28 SUM2=SUM2+PSI1(I)*PSI1(I)                                           POWE0151
      DO 29  I=1,N                                                        POWE0152
      SUM1=SUM1+PSI2(I)*PHI(2,I)                                          POWE0153
   29 SUM2=SUM2+PSI2(I)*PSI2(I)                                           POWE0154
      LAMDA=SUM2/SUM1                                                     POWE0155
      LAMSTR(ICOUT)=LAMDA                                                 POWE0156
      ERRLAM=DABS(LAMDA-LAMB4)                                            POWE0157
C        PUT PSI1 AND PSI2 INTO BIGGER PSI:                               POWE0158
      DO 30  I=1,N                                                        POWE0159
      PSI(1,I)=PSI1(I)                                                    POWE0160
   30 PSI(2,I)=PSI2(I)                                                    POWE0161
C        POINT BY POINT SIMULTANEOUS RELAXATION FLUX ITERATION:           POWE0162
      X=ALPHA                                                             POWE0163
C        DO NOT RELAX DURING THE FIRST THREE ITERATIONS:                  POWE0164
      IF (ICOUT.LE.3)  X=1.0                                              POWE0165
C        CALCULATE THE NEW GROUP FLUX ITERATES AND GROUP ERRORS:          POWE0166
      DO 40  IG=1,2                                                       POWE0167
      DO 40  I=1,N                                                        POWE0168
      PSI(IG,I)=PHI(IG,I) + X*(PSI(IG,I)/LAMDA-PHI(IG,I))                 POWE0169
   40 CONTINUE                                                            POWE0170
C        NORMALIZE THE FLUX ITERATE (ONE GROUP):                          POWE0171
      CALL NORMAL(PSI,N)                                                  POWE0172
      NN=N                                                                POWE0173
      IF (IBC.EQ.1)  NN=N-2                                               POWE0174
C        NORMALIZED ERRORS OF THE FLUX ONLY:                              POWE0175
      DO 39  IG=1,2                                                       POWE0176
      ERRMAX(IG)=0.0                                                      POWE0177
      SQ(IG)=0.0                                                          POWE0178
      IF (IBC.NE.4)  GO TO 37                                             POWE0179
      ERROR(IG,1)=DABS((PSI(IG,1)-PHI(IG,1))/PSI(IG,1) )                  POWE0180
```

```
      ERRMAX(IG)=ERROR(IG,1)
      SQ(IG)=ERROR(IG,1)**2
   37 DO 38  I=2,NN,2
      ERROR(IG,I)=DABS((PSI(IG,I)-PHI(IG,I))/PSI(IG,I) )
      IF (ERROR(IG,I).GT.ERRMAX(IG))  ERRMAX(IG)=ERROR(IG,I)
   38 SQ(IG)=SQ(IG)+ERROR(IG,I)**2
      SQ(IG)=DSQRT(SQ(IG))
C         UPDATE THE FLUX ITERATE:
      DO 39  I=1,N
   39 PHI(IG,I)=PSI(IG,I)
      IF (IFT.EQ.0)  GO TO 31
      DLAM=LAMDA-TRULAM
      DO 36  IG=1,2
      DPHI(IG)=0.0
      DO 35  I=1,N
   35 DPHI(IG)=DPHI(IG)+(PSI(IG,I)-TRUPHI(IG,I))**2
   36 DPHI(IG)=DSQRT(DPHI(IG))
   31 IF (IPLOT.NE.2)  GO TO 45
C         THE FOLLOWING IS FOR NICELY PLOTTING THE GROUP FLUX HISTORY.
          CALL NORM2(PSI,TRUPHI,N)
          K=0
          IF (IBC.EQ.4)  K=1
          KBC=KR
          IF (IBC.EQ.1)  KBC=KR-1
          IF (IBC.EQ.4)  KBC=KR+1
          DO 41  IG=1,2
          DO 41  I=1,KR
          II=I+K
   41     ERROR(IG,II)=PSI(IG,2*I)
          IF (IBC.EQ.4)  ERROR( 1,1)=PSI( 1,1)
          IF (IBC.EQ.4)  ERROR( 2,1)=PSI( 2,1)
C             ERROR NOW CONTAINS THE NEW NORMALIZED FLUX ITERATE PHI.
          JK=IK
          IF (IK.EQ.0)  JK=5
          DO 42  IG=1,2
          DO 42  I=1,KBC
```

```
      IF (DABS(ERROR(IG,I)-PHISTR(IG,I,JK+1)).GE.0.01)  GO TO 43
   42    CONTINUE
C           FLUX HAS NOT CHANGED ENOUGH FOR PLOTTING.
         GO TO 45
C           SAVE THE NORMALIZED FLUX FOR PLOTTING:
   43    IK=IK+1
         IN(IK)=ICOUT
         TE3(IK)=ERRLAM
         DO 44  IG=1,2
         TE1(IG,IK)=ERRMAX(IG)
         TE2(IG,IK)=SQ(IG)
         DO 44  I=1,KBC
   44    PHISTR(IG,I,IK+1)=ERROR(IG,I)
         IF (IK.NE.5)  GO TO 45
C           PLOT THE LAST FIVE SAVED FLUXES:
         CALL PHIPLT(5)
         IK=0
   45 CONTINUE
C        ERROR CRITERIA FOR ACCEPTANCE OF CONVERGENCE.
      IFLAG1=0
      IFLAG2=0
      IFLAG3=0
C        STORE THE ERRORS FOR COMPARISON:
C             ERROR BETWEEN ITERATION EIGENVALUES:
      ERLAM(ICOUT)=ERRLAM
      DO 46  IG=1,2
C             MAXIMUM ERROR BETWEEN ITERATION FLUXES:
      EFSTR(IG,ICOUT)=ERRMAX(IG)
C             MEAN SQUARE ERROR BETWEEN ITERATION FLUXES:
      EFMSTR(IG,ICOUT)=SQ(IG)
C             MEAN SQUARE ERROR BETWEEN THE ITERATION FLUX AND GIVEN TRUE FLUX:
      PHICON(IG,ICOUT)=DPHI(IG)
   46 CONTINUE
C             ERROR BETWEEN THE ITERATION EIGENVALUE AND GIVEN TRUE EIGENVALUE:
      LAMCON(ICOUT)=DLAM
      IF ((ERRMAX(1).LT.EPS1).AND.(ERRMAX(2).LT.EPS1))   IFLAG1=1
```

```
      IF ((SQ(1).LT.EPS2).AND.(SQ(2).LT.EPS2))             IFLAG2=1
      IF (ERRLAM.LT.EPS3)  IFLAG3=1
      IFLAG4=IFLAG1*IFLAG2*IFLAG3
      IF (IFLAG4.EQ.1)  GO TO 50
C        OTHERWISE CONTINUE THE ITERATION.
      LAMB4=LAMDA
      GO TO 20
   50 CONTINUE
C        CONVERGENCE ACCOMPLISHED.
C
C        NORMALIZE THE CONVERGED FLUX VECTOR:
      CALL NORMAL(PHI,N)
C        PLOT ANY LEFT OVER FLUX HISTORY PLOTS:
      IF ((IPLOT.EQ.2).AND.(IK.NE.0))  CALL PHIPLT(IK)
C        BOUNDRY CONDITION INSERTIONS.
      IER=0
C        IER ALLOWS B.C. INSERTIONS FOR YES AND NO CONVERGENCE:
   55 DO 70  IG=1,2
      PHI(IG,N+2)=0.0
      DO 60  I=1,N
      J=N+1-I
   60 PHI(IG,J+1)=PHI(IG,J)
      IF (IBC.NE.4)  GO TO 65
      PHI(IG,2)=0.0
      GO TO 70
   65 PHI(IG,1)=0.0
      IF (IBC.NE.1)  GO TO 70
      PHI(IG,N+2)=PHI(IG,N+1)
      PHI(IG,N+1)=0.0
   70 CONTINUE
   90 IF (IER.EQ.1)  GO TO 102
      RETURN
C        NO CONVERGENCE ACCOMPLISHED:
  100 CONTINUE
C        NORMALIZE THE UNCONVERGED FLUX:
      CALL NORMAL(PHI,N)
```

```
      ICOUT=ICOUT-1
      WRITE (6,101) ICOUT
  101 FORMAT (1H1,'POWER METHOD DID NOT CCNVERGE FOR THIS CASE AFTER',
     X   I4,' ITERATIONS.',//,1X,'EXECUTICN TERMINATED ')
      IER=1
      GO TO 55
  102 CONTINUE
C        FOR PRINTING OUT THE EIGENVALUE HISTORY AND THE FINAL FLUX SHAPE:
      IPLOT=1
      JPLOT=1
      RETURN
      END
```

```
      SUBROUTINE VPROD(A,X,S,N)
C        FORMS THE VECTOR S = PRODUCT OF NXN MATRIX A AND VECTOR X:
C        WHERE A IS THE CUBIC HERMITE (I,6) STORAGE MATRIX.
      REAL*8 A(50,6),X(50),S(50)
      DO 1  I=1,N
      S(I)=0.0
      DO 1  M=1,6
      K=2*(I/2)+M-3
      IF ((K.LT.1).OR.(K.GT.N))  GO TO 1
      S(I)=S(I)+A(I,M)*X(K)
    1 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE SOLVE(A,X,Y,N,G,Z)                                     SOLV0001
C     SOLVES A*X = Y USING CHOLESKY'S METHOD OF FACTORAZATION           SOLV0002
C     FOR POSITIVE DEFINITE REAL AND SYMMETRIC MATRICES A.              SOLV0003
C     REFERENCE:  FORSYTHE & MOLER.                                     SOLV0004
C        G AND Z ARE TEMPORARY WORK AREAS.                              SOLV0005
C     MODIFIED FOR THE CRAZY CUBIC HERMITE (I,6) MATRICES:              SOLV0006
      IMPLICIT REAL*8 (A-H,O-Z)                                         SOLV0007
      DIMENSION A(50,6),G(50,6),X(50),Y(50),Z(50)                       SOLV0008
C        FORM THE MATRIX FACTORAZATION TO G:                            SOLV0009
      CALL FORMG(A,G,N)                                                 SOLV0010
C        SOLVE: G*Z = Y:                                                SOLV0011
      CALL LOWTRI(G,Z,Y,N)                                              SOLV0012
C        FORM G AS SYMMETRIC MATRIX:                                    SOLV0013
      CALL SYMG(G,N)                                                    SOLV0014
C        SOLVE: G-TRANSPOSE*X = Z:                                      SOLV0015
      CALL UPPTRI(G,X,Z,N)                                              SOLV0016
      RETURN                                                            SOLV0017
      END                                                               SOLV0018
```

```
      SUBROUTINE FORMG(A,G,N)                                           FORM0001
C        FORMS MATRIX G FROM A:                                         FORM0002
      IMPLICIT REAL*8 (A-H,O-Z)                                         FORM0003
      DIMENSION A(50,6),G(50,6)                                         FORM0004
      DO 20  J=1,N                                                      FORM0005
      K=3+MOD(J,2)                                                      FORM0006
      K0=K-1                                                            FORM0007
      L0=1                                                              FORM0008
      IF (J.LE.3)  L0=2                                                 FORM0009
      IF (J.EQ.1)  L0=4                                                 FORM0010
      SUM=0.0                                                           FORM0011
      IF (L0.GT.K0)  GO TO 2                                            FORM0012
      DO 1  L=L0,K0                                                     FORM0013
    1 SUM=SUM+G(J,L)**2                                                 FORM0014
    2 SUM=A(J,K)-SUM                                                    FORM0015
      IF (SUM.LT.0.0)  GO TO 100                                        FORM0016
      G(J,K)=DSQRT(SUM)                                                 FORM0017
      IF (J.EQ.N)  GO TO 20                                             FORM0018
      I1=J+1                                                            FORM0019
      I2=J+6-K                                                          FORM0020
      IF (I2.GT.N)  I2=N                                                FORM0021
      M=2                                                               FORM0022
      IF (K.EQ.3)  M=3                                                  FORM0023
      DO 10  I=I1,I2                                                    FORM0024
      SUM=0.0                                                           FORM0025
      L0=1                                                              FORM0026
      IF (I.LE.3)  L0=2                                                 FORM0027
      M0=M-1                                                            FORM0028
      IF (L0.GT.M0)  GO TO 7                                            FORM0029
      DO 5  L=L0,M0                                                     FORM0030
      JL=L                                                              FORM0031
      IF (M.EQ.2)  JL=3                                                 FORM0032
    5 SUM=SUM+G(I,L)*G(J,JL)                                            FORM0033
    7 G(I,M)=(A(I,M)-SUM)/G(J,K)                                        FORM0034
      IF (M.EQ.3)  M=1                                                  FORM0035
   10 CONTINUE                                                          FORM0036
```

```
 20 CONTINUE                                                                FORM0037
    RETURN                                                                  FORM0038
100 WRITE (6,101) J,K                                                       FORM0039
101 FORMAT ('0ERROR IN FORMG:',//,                                          FORM0040
   X   5X,'A(',I2,',',I2,') < 0.0',//,                                      FORM0041
   X   5X,'CHOLESKY METHOD HAS FAILED.',//,                                 FORM0042
   X   5X,'MATRIX A MAY NOT BE POSITIVE DEFINITE OR SYMMETRIC ',//,         FORM0043
   X   '0EXECUTION TERMINATED ')                                            FORM0044
    CALL EXIT                                                               FORM0045
    RETURN                                                                  FORM0046
    END                                                                     FORM0047
```

```
      SUBROUTINE SYMG(G,N)
C        FORMS SYMMETRIC G FROM G LOWER TRIANGULAR:
      REAL*8 G(50,6)
      N2=N-2
C        FILL THE UPPER PORTION OF SYMMETRIC G:
   10 DO 20  I=1,N2
      IF (MOD(I,2).EQ.1)  GO TO 15
C        I IS EVEN:
      G(I,4)=G(I+1,3)
      G(I,5)=G(I+2,1)
      G(I,6)=G(I+3,1)
      GO TO 20
C        I IS ODD:
   15 G(I,5)=G(I+1,2)
      G(I,6)=G(I+2,2)
   20 CONTINUE
      G(N-1,5)=G(N,2)
      RETURN
      END
```

```
      SUBROUTINE LOWTRI(G,Z,Y,N)
C        SOLVES:  G*Z = Y;  FOR Z
C        WHERE G IS NXN LOWER TRIANGULAR.
      REAL*8 G(50,6),Z(50),Y(50),SUM
      DO 10  I=1,N
      K=3+MOD(I,2)
      KO=K-1
      SUM=0.0
      LT=2*(I/2)
      DO 5  M=1,KO
      L=LT+M-3
      IF ((L.LT.1).OR.(L.GT.N))  GO TO 5
      SUM=SUM+G(I,M)*Z(L)
    5 CONTINUE
   10 Z(I)=(Y(I)-SUM)/G(I,K)
      RETURN
      END
```

```
      SUBROUTINE UPPTRI(G,X,Z,N)                                        UPPT0001
C        SOLVES:  G*X = Z;  FOR X                                       UPPT0002
C        WHERE G IS NXN UPPER TRIANULAR.                                UPPT0003
      REAL*8 G(50,6),X(50),Z(50),SUM                                    UPPT0004
      DO 10  J=1,N                                                      UPPT0005
      I=N+1-J                                                           UPPT0006
      K=3+MOD(I,2)                                                      UPPT0007
      K1=K+1                                                            UPPT0008
      SUM=0.0                                                           UPPT0009
      LT=2*(I/2)                                                        UPPT0010
      DO 5  M=K1,6                                                      UPPT0011
      L=LT+M-3                                                          UPPT0012
      IF ((L.LT.1).OR.(L.GT.N))  GO TO 5                                UPPT0013
      SUM=SUM+G(I,M)*X(L)                                               UPPT0014
    5 CONTINUE                                                          UPPT0015
   10 X(I)=(Z(I)-SUM)/G(I,K)                                            UPPT0016
      RETURN                                                            UPPT0017
      END                                                               UPPT0018
```

```
      SUBROUTINE NORMAL(PHI,N)
C     NORMALIZES THE GROUP FLUXES AND CURRENTS BY THE LARGEST FLUX VALUE:
      COMMON /B1/ IBC
      REAL*8 PHI(2,52), A
      NN=N
      IF (IBC.EQ.1)  NN=N-2
      A=0.0
      IF (IBC.NE.4)  GO TO 5
      A=DABS(PHI(1,1))
      IF (DABS(PHI(2,1)).GT.A)  A=DABS(PHI(2,1))
    5 DO 1  IG=1,2
      DO 1  I=2,NN,2
      IF (DABS(PHI(IG,I)).GT.A)  A=DABS(PHI(IG,I))
    1 CONTINUE
      DO 2  IG=1,2
      DO 2  I=1,N
    2 PHI(IG,I)=PHI(IG,I)/A
      RETURN
      END
```

```
      SUBROUTINE PHIPLT(L)
C        PLOTS THE GROUP FLUX HISTORY, WITH UP TO 5 GROUP FLUXES PER PLOT.
C        FAST AND THERMAL GROUP FLUXES ARE PLOTTED SEPERATELY.
C        L IS THE NUMBER OF FLUXES TO BE PLOTTED.
C        L IS BETWEEN 1 AND 5.
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON /B1/ IBC
      COMMON /B2/ KR,N
      COMMON /B5/ A(26,6), B(26,6)
      COMMON /B6/ TE1(2,5),TE2(2,5),TE3(5),IN(5)
      COMMON /ER/ EPS1,EPS2,EPS3
      COMMON /FSTR/ PHISTR(2,26,6)
      DIMENSION SYMBOL(5)
      INTEGER SYMBOL /'.','-','+','#','*'/
      KR1=KR+1
C        SET UP B.C. CONDITIONS
      IF (IBC.EQ.4)  GO TO 5
      IF (IBC.EQ.3)  GO TO 3
      DO 2  IG=1,2
      DO 2  K=1,L
      DO 1  I=1,KR
      J=KR+1-I
    1 PHISTR(IG,J+1,K+1)=PHISTR(IG,J,K+1)
    2 PHISTR(IG,1,K+1)=0.
    3 IF (IBC.EQ.2)  GO TO 5
      DO 4  IG=1,2
      DO 4  K=1,L
    4 PHISTR(IG,KR1,K+1)=0.
    5 CONTINUE
C        FLUXES IN PHISTR HAVE BEEN NORMALIZED IN POWER.
C        PUT THE FAST FLUX IN A, AND THE THERMAL FLUX IN B:
      L1=L+1
      DO 10  K=1,L1
      DO 10  I=1,KR1
      A(I,K)=PHISTR(1,I,K)
   10 B(I,K)=PHISTR(2,I,K)
```

```
C        PLOT THE L FAST FLUX SHAPES ON ONE GRAPH:
      CALL PRTPLT(0,A,KR1,L1,KR1,0,26,6,2)
      WRITE (6,20)
   20 FORMAT (/,'0FAST FLUX ITERATION HISTORY PLOT.',/)
      WRITE (6,30)
   30 FORMAT (
     X   '0KEY:',5X,'SYMBOL',5X,'ITERATION NUMBER:',7X,'ERROR CRITERIA',
     X   11X,'ERROR',13X,'TOLERANCE')
      DO 35  I=1,L
   35 WRITE (6,40) SYMBOL(I),IN(I),TE1(1,I),EPS1,TE2(1,I),EPS2,
     X             TE3(I),EPS3
   40 FORMAT (/,12X,A1,15X,I3,16X,'FLUX',14X,1PD15.5,5X,1PD15.5,/,
     X   47X,'MEAN SQ. FLUX',5X,1PD15.5,5X,1PD15.5,/,
     X   47X,'EIGENVALUE',8X,1PD15.5,5X,1PD15.5)
C        PLOT THE L THERMAL FLUX SHAPES ON THE OTHER GRAPH:
      CALL PRTPLT(0,B,KR1,L1,KR1,0,26,6,2)
      WRITE (6,50)
   50 FORMAT (/,'0THERMAL FLUX ITERATION PLOT.',/)
      WRITE (6,30)
      DO 55  I=1,L
   55 WRITE (6,40) SYMBOL(I),IN(I),TE1(2,I),EPS1,TE2(2,I),EPS2,
     X             TE3(I),EPS3
      RETURN
      END
```

```
      SUBROUTINE CURENT                                                 CURE0001
C        FORMS THE SEPERATE FLUX AND CURRENT VECTORS                    CURE0002
C        FROM THE COMBINED ELEMENTS OF PHI.                             CURE0003
C        THEN:  FLUX=PHI;  CURRENT=CUR.                                 CURE0004
      COMMON /B2/ KR,N                                                  CURE0005
      COMMON /B4/ PHI(2,52),CUR(2,52)                                   CURE0006
      REAL*8 PHI,CUR                                                    CURE0007
      KR1=KR+1                                                          CURE0008
      DO 2  IG=1,2                                                      CURE0009
      DO 1  K=1,KR1                                                     CURE0010
    1 CUR(IG,K)=PHI(IG,2*K)                                             CURE0011
      DO 2  K=1,KR1                                                     CURE0012
    2 PHI(IG,K)=PHI(IG,2*K-1)                                           CURE0013
      RETURN                                                            CURE0014
      END                                                               CURE0015
```

```
      SUBROUTINE OUTPUT
C        PRINTS THE RESULTS OF THE METHOD.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH
      COMMON /B2/ KR,N
      COMMON /B4/ PHI(2,52), CUR(2,52), LAMDA, ICOUT
      COMMON /B5/ PSI(2,26)
      COMMON /ER/ EPS1,EPS2,EPS3
      COMMON /ESTR/ LAMSTR(300),EFSTR(2,300),EFMSTR(2,300),ERLAM(300)
      COMMON /TRUE/ TRULAM, TRUPHI(2,52), PHICON(2,300), LAMCON(300),IFT
      INTEGER N
      KR0=KR-1
      KR1=KR+1
      WRITE (6,1)
    1 FORMAT ('1RESULTS OF THE MULTIGROUP METHOD:')
      WRITE (6,10) ICOUT
   10 FORMAT (//,' PROBLEM TERMINATED AFTER',I5,
     X        ' OUTER (POWER) ITERATIONS TO:')
      WRITE (6,20) LAMDA
   20 FORMAT (/,10X,'LAMDA = ',1PE21.14)
C        PRINT OUT EIGENVALUES.
      CALL PLOT
      WRITE (6,30)
   30 FORMAT ('1RESULTS AFTER PROBLEM TERMINATION:',/,
     X   '0INDEX',8X,'THERMAL FLUX',11X,'FAST FLUX',5X,
     X   'THERMAL CURRENT',8X,'FAST CURRENT',/)
      WRITE (6,50) (K,PHI(2,K),PHI(1,K),CUR(2,K),CUR(1,K),K=1,KR1)
   50 FORMAT (I6,4D20.7)
      IF (IPUNCH.EQ.1)  CALL PUNCH
C        PRINT OUT GROUP NORMALIZED RESULTS:
      DO 52  IG=1,2
      A=0.0
      DO 51  I=1,KR1
      IF (PHI(IG,I).GT.A)  A=PHI(IG,I)
   51 CONTINUE
      DO 52  I=1,KR1
```

```
      PHI(IG,I)=PHI(IG,I)/A
   52 CUR(IG,I)=CUR(IG,I)/A
      WRITE (6,55) (K,PHI(2,K),PHI(1,K),CUR(2,K),CUR(1,K),K=1,KR1)
   55 FORMAT (//,'0GROUP NORMALIZED RESULTS:',//,(I6,4D20.7))
C         CALCULATE THE FINAL TO EXPECTED FLUX RATIOS:
      K1=1
      K2=KR1
      IF (IBC.LE.2)  K1=2
      IF ((IBC.EQ.1).OR.(IBC.EQ.3))  K2=KR
      DO 60  IG=1,2
      IF (IBC.LE.2)  PSI(IG,1)=1.0
      IF ((IBC.EQ.1).OR.(IBC.EQ.3))  PSI(IG,KR1) = 1.0
      I=0
      DO 60  K=K1,K2
      I=I+2
   60 PSI(IG,K)=PHI(IG,K)/TRUPHI(IG,I)
      WRITE (6,70) (I,PSI(2,I),PSI(1,I),I=1,KR1)
   70 FORMAT ('1RATIOS OF THE TERMINATED GROUP FLUX TO THE EXPECTED GROU
     XP FLUX:',//,
     X   10X,'- AN INDICATION OF THE ACCURACY OF THE CONVERGENCE -',///,
     X   '    K',12X,'THERMAL RATIO',15X,'FAST RATIO',//,(I5,2E25.10))
C         PRINT OUT THE STORED ITERATION ERRORS:
      WRITE (6,110) EPS1,(EFSTR(2,I),I=1,ICOUT)
      WRITE (6,111) EPS1,(EFSTR(1,I),I=1,ICOUT)
      WRITE (6,112) EPS2,(EFMSTR(2,I),I=1,ICOUT)
      WRITE (6,113) EPS3,(EFMSTR(1,I),I=1,ICOUT)
      WRITE (6,114) EPS3,(ERLAM(I),I=1,ICOUT)
  110 FORMAT ('1MAXIMUM NORMALIZED ERRORS BETWEEN THE THERMAL FLUX ITERA
     XTIONS:',
     X   25X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  111 FORMAT ('1MAXIMUM NORMALIZED ERRORS BETWEEN THE FAST FLUX ITERATIO
     XNS:',
     X   25X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  112 FORMAT ('1MEAN SQUARE NORMALIZED ERROR BETWEEN THE THERMAL FLUX IT
     XERATIONS:',
     X   18X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
```

```
  113 FORMAT ('1MEAN SQUARE NORMALIZED ERROR BETWEEN THE FAST FLUX ITERA
     XTIONS:',
     X    18X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
  114 FORMAT ('1ERROR BETWEEN THE ITERATICN EIGENVALUES:',
     X    28X,'TOLERANCE USED = ',1PE12.4,//, (1P5E20.5))
      IF (IFT.EQ.0)  RETURN
C        PRINT OUT THE GIVEN TRUE EIGENVALUE AND FLUX:
      WRITE (6,115) TRULAM,((TRUPHI(3-J,I),J=1,2),I=1,N)
  115 FORMAT ('1THE GIVEN TRUE EIGENVALUE:',//,15X,
     X    'TRULAM =',E22.14,///,
     X    '0THE GIVEN MULTIGROUP FLUXES:',//,
     X    13X,'THERMAL',16X,'FAST',//,(2D20.10))
C        PRINT OUT THE STORED CONVERGENCE ERRORS:
      WRITE (6,120) (PHICON(2,I),I=1,ICOUT)
      WRITE (6,121) (PHICON(1,I),I=1,ICOUT)
      WRITE (6,122) (LAMCON(I),I=1,ICCUT)
  120 FORMAT ('1MEAN SQUARE ERROR BETWEEN THE THERMAL ITERATION FLUX AND
     X THE GIVEN TRUE THERMAL FLUX:',//,(1P5E20.5))
  121 FORMAT ('1MEAN SQUARE ERROR BETWEEN THE FAST ITERATICN FLUX AND TH
     XE GIVEN TRUE FAST FLUX:',//,(1P5E20.5))
  122 FORMAT ('1ERROR BETWEEN THE ITERATICN EIGENVALUES AND THE GIVEN TR
     XUE EIGENVALUE:',//,(1P5E20.5))
C
      RETURN
      END
```

```
      SUBROUTINE PLOT
C        PLOTS OUT THE EIGENVALUE HISTORY AS A TABLE AND A GRAPH,
C        AS WELL AS PLOTTING OUT THE FINAL MULTIGROUP FLUX SHAPES.
      IMPLICIT REAL*8 (A-H,L-Z)
      COMMON /B1/ IBC,IPLOT,JPLOT,IPUNCH
      COMMON /B2/ KR
      COMMON /B4/ PHI(2,52), PSI(2,52), LAMDA, ICOUT
      COMMON /B5/ B(300,2)
      COMMON /ESTR/ LAMSTR(300)
      DIMENSION C(26,3)
C        IN ORDER TO SAVE SOME SPACE:
      EQUIVALENCE (B(1),C(1))
      WRITE (6,1) (LAMSTR(I),I=1,ICOUT)
    1 FORMAT ('OTABLE OF EIGENVALUES DURING THE POWER ITERATION:',
     X        //,(1P5E25.14))
      IF (JPLOT.EQ.0)  GO TO 20
      DO 10  I=1,ICOUT
      B(I,1)=I
   10 B(I,2)=LAMSTR(I)
      CALL PRTPLT(1,B,ICOUT,2,ICOUT,0,300,2,1)
      WRITE (6,11)
   11 FORMAT ('OPLOT OF THE EIGENVALUE HISTORY THROUGH THE ITERATIONS.')
   20 IF (IPLOT.EQ.0)  RETURN
      KR1=KR+1
      DO 30  I=1,KR1
      C(I,1)=I
      C(I,2)=PHI(1,I)
   30 C(I,3)=PHI(2,I)
      CALL PRTPLT(2,C,KR1,3,KR1,0,26,3,2)
      WRITE (6,31)
   31 FORMAT ('OFINAL CCNVERGED CCNNECTING FLUX POINTS; F(K).',//,
     X   5X,'FAST FLJX:     .',/,5X,'THERMAL FLUX:  -')
      RETURN
      END
```

```
      SUBROUTINE PUNCH                                                  PNCH0001
C        PUNCHES OUT THE CUBIC RESULTS FOR PLOT2G ROUTINE INPUT:        PNCH0002
      REAL*8 F,C                                                        PNCH0003
      COMMON /B2/ KR                                                    PNCH0004
      COMMON /B4/ F(2,52), C(2,52)                                      PNCH0005
      KR1=KR+1                                                          PNCH0006
      WRITE (7,1) KR,(F(1,I),C(1,I),F(2,I),C(2,I),I=1,KR1)              PNCH0007
    1 FORMAT (I5,/,(4D20.7))                                            PNCH0008
      WRITE (6,100)                                                     PNCH0009
  100 FORMAT (///,' THE OUTPUT HAS BEEN PUNCHED OUT ONTO CARDS ')       PNCH0010
      RETURN                                                            PNCH0011
      END                                                               PNCH0012
```

```
      SUBROUTINE NORM2(PSI,TRUPHI,N)
C     NORMALIZES BOTH ENERGY GROUP FLUXES IN PSI TO 1.0:
C     DITTO FOR TRUPHI ON THE FIRST CALL.
      COMMON /B1/ IBC
      REAL*8 PSI(2,52), TRUPHI(2,52), A(2)
      DATA K /0/
      K=K+1
      NN=N
      IF (IBC.EQ.1)  NN=NN-2
      DO 1  IG=1,2
      A(IG)=0.0
      IF (IBC.NE.4)  GO TO 7
      A(IG)=DABS(PSI(IG,1))
    7 DO 1  I=2,NN,2
      IF (DABS(PSI(IG,I)).GT.A(IG))  A(IG)=DABS(PSI(IG,I))
    1 CONTINUE
      DO 2  IG=1,2
      DO 2  I=1,N
      IF (A(IG).EQ.0.0)  GO TO 2
      PSI(IG,I)=PSI(IG,I)/A(IG)
    2 CONTINUE
      IF (K.NE.1)  RETURN
      DO 5  IG=1,2
      A(IG)=0.
      IF (IBC.NE.4)  GO TO 8
      A(IG)=DABS(TRUPHI(IG,1))
    8 DO 5  I=2,NN,2
      IF (TRUPHI(IG,I).GT.A(IG))  A(IG)=TRUPHI(IG,I)
    5 CONTINUE
      DO 6  IG=1,2
      DO 6  I=1,N
      IF (A(IG).EQ.0.0)  GO TO 6
      TRUPHI(IG,I)=TRUPHI(IG,I)/A(IG)
    6 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PRTPLT(NO,B,N,M,NL,NS,KX,JX,ISP)                       PRTP0001
C                                                                       PRTP0002
C     *  IDENTICAL TO SUBROUTINE PRTPLT PREVIOUSLY LISTED IN PROGRAM REF2G.  PRTP0003
C                                                                       PRTP0004
      RETURN                                                            PRTP0005
      END                                                               PRTP0006
```

F.4. SOURCE LISTING of Program ANALYZE

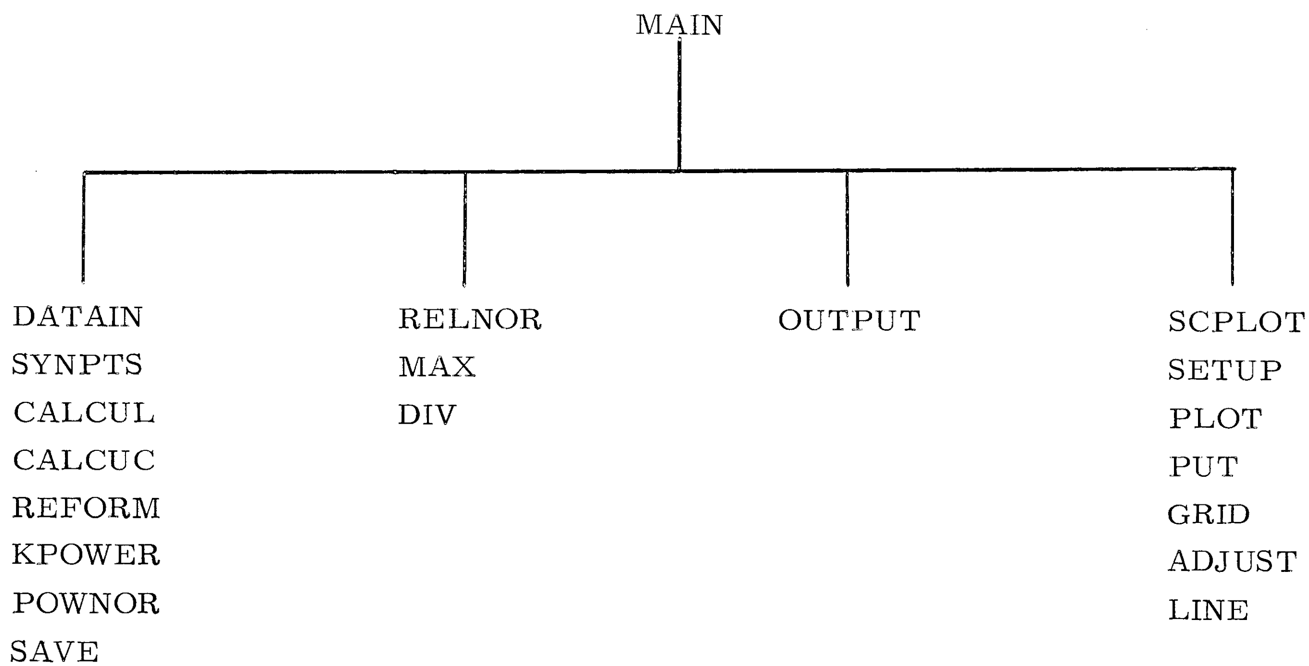


Figure F.4. Structure of Program ANALYZE.
Not including the M.I.T. SC-4020 Subroutine Package.[54]

```
C      PROGRAM ANALYZE:
C      DATA ANALYSIS AND COMPARISON PROGRAM:
       DIMENSION  NP(200),CHI(2),R(9,26),F(2,201),G(2,201),H(1000),
     X            X(1001),PHI(2,1001),SF(2,1000),D(2,1000),
     X            U(2,1001),XB(1001),UB(2,1001),XC(1001),UC(2,1001),
     X            XS(1001),NSR(3,26),NRNK(25)
C         THETA IS DEFINED SUCH THAT THE CURRENT (0) = THETA * G, FOR EACH K.
C         DO FOR THE 3 FLUX DATA BLOCKS:
      DO 10  IT=1,3
C         READ IN THE MATERIALS INPUT DATA BLOCK:
      CALL DATAIN(IT,METHOD,NK,NR,NRNK,NP,N,XS,PHI,SF,D,CHI,THETA,NAP,H)
C         READ IN THE CONVERGED FLUX POINTS DATA BLOCK:
      CALL SYNPTS(IT,METHOD,NK,NR,F,G)
C         CALCULATE THE DETAILED FLUX SHAPES:
      IF (METHOD.EQ.2)  GO TO 4
      CALL CALCUL(IT,METHOD,NK,NR,NRNK,NP,N,X,U,XS,PHI,SF,D,CHI,
     X            THETA,F,G)
      GO TO 5
    4 CALL CALCUC(IT,METHOD,NK,NR,NRNK,NP,N,X,U,XS,PHI,SF,D,CHI,
     X            THETA,F,G,NAP)
C         TRANSFORM THE FLUX POINTS INTO NK DISTINCT REGIONS:
    5 CALL REFORM(IT,METHOD,NK,NR,NRNK,NP,N,X,U,SF,D)
C         CALCULATE THE POWER IN EACH OF THE NK REGIONS FOR EACH FLUX:
      CALL KPOWER(IT,METHOD,NK,NP,N,X,U,SF,D,CHI,THETA,NAP,F,G,H,
     X            NSR,R,TPOWER)
C         NORMALIZE THE REGION POWERS AND THE FLUX POINTS BY TPOWER)
      CALL POWNOR(IT,METHOD,NK,NP,N,X,U,R,TPOWER)
C         SAVE THESE RESULTS FOR PLOTTING:
      CALL SAVE(IT,NK,N,X,U,NB,XB,UB,NC,XC,UC)
   10 CONTINUE
C         NORMALIZE THE FLUX RESULTS TOGETHER FOR EACH GROUP TO 1.0:
      CALL RELNOR(NK,N,X,U,NB,XB,UB,NC,XC,UC)
C         PRINT OUT THE POWER AND ARRAY RESULTS:
      CALL OUTPUT(NK,N,X,U,NB,XB,UB,NC,XC,UC,NSR,R)
C         READ IN THE PLOTTING INFORMATION AND PLOT THE RESULTS ON THE SC 4020:
      CALL SCPLOT(NK,N,X,U,NB,XB,UB,NC,XC,UC)
```

```
      WRITE (6,20)                                          ANAL0037
   20 FORMAT (/,'ONORMAL PROGRAM TERMINATION.')             ANAL0038
      STOP                                                  ANAL0039
      END                                                   ANAL0040
```

```
      SUBROUTINE DATAIN(IT,METHOD,NK,NR,NRNK,NP,N,X,PHI,SF,D,CHI,THETA,
     X                 NAP,H)
C        READS IN THE MATERIAL DATA INPUT BLOCKS:
      DIMENSION  NP(200),K(200),L(200),ITF(200),KTF(200),CHI(2),
     X           X(1001),PHI(2,1001),SF(2,1000),D(2,1000),H(1000),
     X           NRNK(25)
C        DATA BLOCK ID CARDS:
      READ (IT,20) METHOD, NK, NR, NAP
C        NRNK = NUMBER OF NR REGIONS PER EACH NK REGION:
      IF (NR.NE.NK)  READ (IT,20) (NRNK(I),I=1,NK)
   20 FORMAT (16I5)
      XSTART=0.0
      K(1)=1
      READ (IT,11) NR
      READ (IT,12) (ITF(I),I=1,NR)
      READ (IT,17) CHI(1),CHI(2), THETA
      IF (THETA.EQ.0.)  THETA=1.
      NUMITF=1
      DO 10  I=1,NR
      IF (ITF(I).LT.NUMITF)  GO TO 2
C        NEW REGION DATA:
      NUMITF=NUMITF+1
      KTF(NUMITF-1)=I
      READ (IT,13) NS
      IF (NS.EQ.0)  NS=1
      NP(I)=NS+1
      IF (I.NE.1)  K(I)=K(I-1)+NP(I-1)
      L(I)=NP(I)
      IF (I.NE.1)  L(I)=L(I-1)+NP(I)
      K0=K(I)
      L0=L(I)
      L1=L0-1
      L2=L0-2
      IF (NS.GT.1)  READ (IT,14) (X(J),H(J),SF(1,J),D(1,J),
     X                          SF(2,J),D(2,J),J=K0,L2)
      READ (IT,15) X(L1),X(L0),H(L1),SF(1,L1),D(1,L1),SF(2,L1),D(2,L1)
```

```
      XSTART=XSTART-X(KO)                                DATA0037
      DO 1  J=KO,LO                                      DATA0038
    1 X(J)=X(J)+XSTART                                   DATA0039
      XSTART=X(LO)                                       DATA0040
      IF (IT.EQ.3)  GO TO 10                             DATA0041
      READ (IT,16) (PHI(1,J),J=KO,LO)                    DATA0042
      READ (IT,16) (PHI(2,J),J=KO,LO)                    DATA0043
      GO TO 10                                           DATA0044
C        OLD REPEATED REGION DATA:                       DATA0045
    2 M=ITF(I)                                           DATA0046
      M=KTF(M)                                           DATA0047
      NP(I)=NP(M)                                        DATA0048
      K(I)=K(I-1)+NP(I-1)                                DATA0049
      L(I)=L(I-1)+NP(I)                                  DATA0050
      KO=K(I)                                            DATA0051
      LO=L(I)                                            DATA0052
      L1=LO-1                                            DATA0053
      KM=K(M)                                            DATA0054
      IB=KO-KM                                           DATA0055
      XSTART=XSTART-X(KM)                                DATA0056
      DO 5  J=KO,LO                                      DATA0057
      NJ=J-IB                                            DATA0058
      X(J)=X(NJ)+XSTART                                  DATA0059
      H(J)=H(NJ)                                         DATA0060
      DO 5  IG=1,2                                       DATA0061
      D(IG,J)=D(IG,NJ)                                   DATA0062
      SF(IG,J)=SF(IG,NJ)                                 DATA0063
      PHI(IG,J)=PHI(IG,NJ)                               DATA0064
    5 CONTINUE                                           DATA0065
      XSTART=X(LO)                                       DATA0066
   10 CONTINUE                                           DATA0067
   11 FORMAT (/,I5)                                      DATA0068
   12 FORMAT (25I2)                                      DATA0069
   13 FORMAT (5X,I5)                                     DATA0070
   14 FORMAT (F10.5,10X,F10.5,10X,2E10.3,/,40X,2E10.3)   DATA0071
   15 FORMAT (3F10.5,10X,2E10.3,/,40X,2E10.3)            DATA0072
```

```
16 FORMAT (E20.7)
17 FORMAT (3F10.5)
   N=L(NR)
   IF (IT.NE.3)  RETURN
   DO 50  I=1,N
   DO 50  IG=1,2
50 PHI(IG,I)=1.0
   RETURN
   END
```

```
      SUBROUTINE SYNPTS(IT,METHOD,NK,NR,F,G)                            SYNP0001
C     READS IN CONVERGED FLUX AND CURRENT POINTS:                       SYNP0002
      DIMENSION F(2,201),G(2,201)                                       SYNP0003
      JT=IT+10                                                          SYNP0004
      READ (JT,1) NR                                                    SYNP0005
    1 FORMAT (I5)                                                       SYNP0006
      NR1=NR+1                                                          SYNP0007
      IF (METHOD.NE.1)  GO TO 10                                        SYNP0008
C        FOR LINEAR SYNTHESIS:                                          SYNP0009
      IF (IT.NE.3)  READ (JT,2) (F(1,I),F(2,I),I=1,NR1)                 SYNP0010
      IF (IT.EQ.3)  READ (JT,3) (F(1,I),I=1,NR1),(F(2,I),I=1,NR1)       SYNP0011
    2 FORMAT (2E20.7)                                                   SYNP0012
    3 FORMAT (E20.7)                                                    SYNP0013
      GO TO 20                                                          SYNP0014
   10 READ (JT,11) (F(1,I),G(1,I),F(2,I),G(2,I),I=1,NR1)                SYNP0015
   11 FORMAT (4E20.7)                                                   SYNP0016
   20 RETURN                                                            SYNP0017
      END                                                               SYNP0018
```

```
      SUBROUTINE CALCUL(IT,METHOD,NK,NR,NRNK,NP,NU,XU,U,XS,PHI,SF,D,CHI,
     X                  THETA,F,G)
C        CALCULATES THE FLUX TRIAL FUNCTICN U FOR LINEAR SYNTHESIS:
      DIMENSION  NP(200),XU(1001),U(2,1001),PHI(2,1001),SF(2,1000),
     X           D(2,1000),CHI(2),F(2,201),G(2,201),XS(1001),NRNK(25)
      ISYNTH=20
C        FOR EACH K'TH REGION:
      N=0
      LL=0
      DO 20  K=1,NR
      L=LL+1
      LL=LL+NP(K)
      DO 7  IG=1,2
      IF (PHI(IG,L).EQ.0.0)  PHI(IG,L)=1.0E-6
      IF (PHI(IG,LL).EQ.0.0)  PHI(IG,LL)=1.0E-6
    7 CONTINUE
C        FOR ALL POINTS IN THIS REGION:
      DO 10  I=L,LL
      N=N+1
      XU(N)=XS(I)
      X=(XU(N)-XS(L))/(XS(LL)-XS(L))
      IF ((K.EQ.1).AND.(ISYNTH.EQ.21.OR.ISYNTH.EQ.24))  X=1.0
      IF ((K.EQ.NR).AND.(ISYNTH.EQ.22.OR.ISYNTH.EQ.24))  X=0.0
      DO 1  IG=1,2
    1 U(IG,N)=PHI(IG,I)*(F(IG,K)*(1.-X)/PHI(IG,L)
     X                   +F(IG,K+1)*X/PHI(IG,LL))
   10 CONTINUE
   20 CONTINUE
      NU=N
      RETURN
      END
```

```
      SUBROUTINE CALCUC(IT,METHOD,NK,NR,NRNK,NP,NU,XU,U,XS,PHI,SF,D,CHI,
     X                  THETA,F,G,NAPT)
C         CALCULATES THE FLUX TRIAL FUNCTION U FOR CUBIC HERMITE SYNTHESIS:
C         NAP = # OF ADDITIONAL POINTS TO BE CALCULATED IN EACH REGION
C         AND IS NEGATIVE IF NAP APPLIES ONLY TO THE FIRST REGION.
      DIMENSION  NP(200),XU(1001),U(2,1001),PHI(2,1001),SF(2,1000),
     X           D(2,1000),CHI(2),F(2,201),G(2,201),XS(1001),NRNK(25)
      NAP=NAPT
      NL=0
      DO 50  K=1,NR
   50 NL=NL+NP(K)
C         FOR EACH K'TH REGION:
      N=0
      LL=0
      DO 20  K=1,NR
      L=LL+1
      LL=LL+NP(K)
      H=XS(LL)-XS(L)
      DO 7  IG=1,2
      IF (PHI(IG,L).EQ.0.0)  PHI(IG,L)=1.0E-6
      IF (PHI(IG,LL).EQ.0.0)  PHI(IG,LL)=1.0E-6
    7 CONTINUE
C         FOR ALL POINTS IN THIS REGION:
      DO 10  I=L,LL
      N=N+1
      XU(N)=XS(I)
      X=(XU(N)-XS(L))/(XS(LL)-XS(L))
      DO 1  IG=1,2
    1 U(IG,N)=PHI(IG,I)*(F(IG,K)*(1.-3.*X**2+2.*X**3)/PHI(IG,L)
     X   +F(IG,K+1)*(3.*X**2-2.*X**3)/PHI(IG,LL)
     X   +H*(G(IG,K)/(D(IG,L)*PHI(IG,L))*(-X+2.*X**2-X**3)
     X   +G(IG,K+1)/(D(IG,LL-1)*PHI(IG,LL))*(X**2-X**3))*THETA)
      NAPP=IABS(NAP)
      IF (NAP.EQ.0)  GO TO 10
      IF (I.EQ.LL)  GO TO 10
      DX=(XS(I+1)-XS(I))/FLOAT(NAPP+1)
```

```
      DO 5  INP=1,NAPP
      N=N+1
      XU(N)=XU(N-1)+DX
      X=(XU(N)-XS(L))/(XS(LL)-XS(L))
      DO 5  IG=1,2
    5 U(IG,N)=PHI(IG,I)*(F(IG,K)*(1.-3.*X**2+2.*X**3)/PHI(IG,L)
     X    +F(IG,K+1)*(3.*X**2-2.*X**3)/PHI(IG,LL)
     X    +H*(G(IG,K)/(D(IG,L)*PHI(IG,L))*(-X+2.*X**2-X**3)
     X    +G(IG,K+1)/(D(IG,LL-1)*PHI(IG,LL))*(X**2-X**3))*THETA)
   10 CONTINUE
      IF (NAPP.EQ.0)  GO TO 15
      IF (IT.EQ.1)  GO TO 15
C         SHIFTING SF AND D ARRAYS TO COMPENSATE FOR NAP .NE. 0:
      NS=NP(K)-1
      NNP=NS*NAPP
      NPS=0
      K1=K+1
      IF (K1.GT.NR)  GO TO 60
      DO 51  I=K1,NR
   51 NPS=NPS+NP(I)
      DO 55  J=1,NPS
      I=NL+1-J
      II=I+NNP
      DO 55  IG=1,2
      SF(IG,II)=SF(IG,I)
   55 D(IG,II)=D(IG,I)
   60 NL=NL+NNP
      L=NL-NPS
      L1=L-NNP-NP(K)
      DO 70  I=1,NS
      II=NS+1-I+L1
      NAPP1=NAPP+1
      DO 65  J=1,NAPP1
      L=L-1
      DO 65  IG=1,2
      SF(IG,L)=SF(IG,II)
```

```
65 D(IG,L)=D(IG,II)
70 CONTINUE
15 NP(K)=NP(K)+NAPP
   IF (NAP.LT.0)  NAP=0
20 CONTINUE
   NU=N
   RETURN
   END
```

```
      SUBROUTINE REFORM(IT,METHOD,NK,NR,NRNK,NP,N,X,U,SF,D)
C     REFORMS THE NR GIVEN REGIONS INTO NK DESIRED REGIONS:
      DIMENSION  NP(200),X(1001),U(2,1001),SF(2,1000),D(2,1000),NRNK(25)
      IF (NK.EQ.NR)  RETURN
C        IT SHOULD BE 3 ONLY.
C        DELETE ALL DOUBLE ENTRIES WITHIN EACH NK DATA BLOCK:
      M=-1
      I=0
      DO 5  K=1,NK
      L=NRNK(K)
      DO 4  J=1,L
      I=I+1
      M=M+2
      X(I)=X(M)
      DO 4  IG=1,2
      U(IG,I)=U(IG,M)
      SF(IG,I)=SF(IG,M)
    4 D(IG,I)=D(IG,M)
      I=I+1
      X(I)=X(M+1)
      DO 5  IG=1,2
    5 U(IG,I)=U(IG,M+1)
      N=I
C        NP(K) = # OF FLUX POINTS IN REGION K = # NS SUBREGIONS + 1:
      DO 10  K=1,NK
   10 NP(K)=NP(K)+NRNK(K)-1
      NR=NK
      RETURN
      END
```

```
      SUBROUTINE KPOWER(IT,METHOD,NK,NP,N,X,U,SF,D,CHI,THETA,NAP,F,G,           KPOW0001
     X                  H,NSR,R,TPOWER)                                         KPOW0002
C     CALCULATES THE POWER IN EACH REGION BY INTEGRATIONS:                      KPOW0003
      DIMENSION  NP(200),X(1001),U(2,1001),SF(2,1000),D(2,1000),CHI(2),         KPOW0004
     X           F(2,201),G(2,201),H(1000),R(9,26),NSR(3,26)                    KPOW0005
      TPOWER=0.0                                                                KPOW0006
      IF (IT.NE.1)  GO TO 20                                                    KPOW0007
C        FOR THE HOMOGENEOUS CASE:                                              KPOW0008
C        ASSUMING THAT NP IS ALWAYS 2 AND THUS D'S ARE CONSTANT.                KPOW0009
      DO 10  K=1,NK                                                             KPOW0010
      M=2*K-1                                                                   KPOW0011
      NSR(IT,K)=1                                                               KPOW0012
      R(IT,K)=0.0                                                               KPOW0013
      DO 8  IG=1,2                                                              KPOW0014
      IF (METHOD.NE.1)  GO TO 5                                                 KPOW0015
C        LINEAR FLUX:                                                           KPOW0016
      R(IT,K)=(F(IG,K)+F(IG,K+1))*H(M)*SF(IG,M)/2.0+R(IT,K)                     KPOW0017
      GO TO 8                                                                   KPOW0018
C        CUBIC FLUX:                                                            KPOW0019
    5 R(IT,K)=((F(IG,K)+F(IG,K+1))*H(M)/2.0+THETA*(-G(IG,K)                     KPOW0020
     X   /D(IG,M)+G(IG,K+1)/D(IG,M))*H(M)**2/12.0)*SF(IG,M)+R(IT,K)             KPOW0021
    8 CONTINUE                                                                  KPOW0022
   10 TPOWER=TPOWER+R(IT,K)                                                     KPOW0023
      RETURN                                                                    KPOW0024
C        SYNTHESIS OR REFERENCE DETAILED FLUX CASE:                             KPOW0025
   20 M=-1                                                                      KPOW0026
      DO 50  K=1,NK                                                             KPOW0027
      M=M+1                                                                     KPOW0028
      NS=NP(K)-1                                                                KPOW0029
      NSR(IT,K)=NS                                                              KPOW0030
      R(IT,K)=0.0                                                               KPOW0031
      DO 40  J=1,NS                                                             KPOW0032
      M=M+1                                                                     KPOW0033
      DO 40  IG=1,2                                                             KPOW0034
   40 R(IT,K)=(U(IG,M)+U(IG,M+1))*(X(M+1)-X(M))*SF(IG,M)/2.0 + R(IT,K)          KPOW0035
   50 TPOWER=TPOWER+R(IT,K)                                                     KPOW0036
```

```
      RETURN                                                          KPOW0037
      END                                                             KPOW0038
```

```
      SUBROUTINE POWNOR(IT,METHOD,NK,NP,N,X,U,R,TPOWER)                 POWN0001
C     NORMALIZES THE FLUXES AND REGION POWERS BY TPOWER:                POWN0002
      DIMENSION  NP(200),X(1001),U(2,1001),R(9,26)                      POWN0003
      DO 1  K=1,NK                                                      POWN0004
    1 R(IT+3,K)=R(IT,K)/TPOWER                                          POWN0005
      DO 2  IG=1,2                                                      POWN0006
      DO 2  K=1,N                                                       POWN0007
    2 U(IG,K)=U(IG,K)/TPOWER                                            POWN0008
      RETURN                                                            POWN0009
      END                                                               POWN0010
```

```
      SUBROUTINE SAVE(IT,NK,N,X,U,NB,XB,UB,NC,XC,UC)
C     SAVES THE FLUX DISTRIBUTIONS IN THE IT LOOP BY PLACING:
C        IT = 1:  HOMOGENEOUS RESULTS IN UC;
C        IT = 2:  SYNTHESIS RESULTS   IN UB;
C        IT = 3:  REFERENCE RESULTS   IN U.
      DIMENSION  X(1001),U(2,1001),XB(1001),UB(2,1001),
     X           XC(1001),UC(2,1001)
      IF (IT.NE.1)  GO TO 2
C        HOMOGENEOUS RESULTS:
      DO 1  I=1,N
      XC(I)=X(I)
      DO 1  IG=1,2
    1 UC(IG,I)=U(IG,I)
      NC=N
      RETURN
    2 IF (IT.EQ.3)  RETURN
C        SYNTHESIS RESULTS:
      DO 3  I=1,N
      XB(I)=X(I)
      DO 3  IG=1,2
    3 UB(IG,I)=U(IG,I)
      NB=N
      RETURN
      END
```

```
      SUBROUTINE RELNOR(NK,N,X,U,NB,XB,UB,NC,XC,UC)
C     NORMALIZES THE SET OF U, UB, UC; TO UNITY FOR EACH GROUP:
      DIMENSION  X(1001),U(2,1001),XB(1001),UB(2,1001),
     X           XC(1001),UC(2,1001)
      DIMENSION A(2),B(2),C(2),Z(2,3)
      CALL MAX(N,U,A)
      CALL MAX(NB,UB,B)
      CALL MAX(NC,UC,C)
      DO 1  IG=1,2
      Z(IG,1)=A(IG)
      Z(IG,2)=B(IG)
    1 Z(IG,3)=C(IG)
      CALL MAX(3,Z,A)
      CALL DIV(N,U,A)
      CALL DIV(NB,UB,A)
      CALL DIV(NC,UC,A)
      RETURN
      END
```

```
      SUBROUTINE MAX(N,U,A)
C     FINDS THE MAXIMUM POSITIVE ELEMENT CF U FOR EACH GROUP:
      DIMENSION U(2,1001),A(2)
      DO 1  IG=1,2
      A(IG)=0.
      DO 1  I=1,N
      IF (U(IG,I).GT.A(IG))  A(IG)=U(IG,I)
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DIV(N,U,A)                                             DIV 0001
C     DIVIDES A INTO U FOR EACH GROUP:                                  DIV 0002
      DIMENSION U(2,1001),A(2)                                          DIV 0003
      DO 1  IG=1,2                                                      DIV 0004
      DO 1  I=1,N                                                       DIV 0005
    1 U(IG,I)=U(IG,I)/A(IG)                                             DIV 0006
      RETURN                                                            DIV 0007
      END                                                               DIV 0008
```

```
      SUBROUTINE OUTPUT(NK,N,X,U,NB,XB,UB,NC,XC,UC,NSR,R)
C     PRINTS OUT THE ANALYSIS RESULTS:
      DIMENSION  X(1001),U(2,1001),XB(1001),UB(2,1001),
     X           XC(1001),UC(2,1001),R(9,26),NSR(3,26)
C        SUM UP THE REGION RAW POWERS AND REGION FRACTIONAL POWERS:
      DO 5  IT=1,6
      IF (IT.LE.3)  NSR(IT,26)=0
      R(IT,26)=0.0
      DO 5  K=1,NK
      IF (IT.LE.3)  NSR(IT,26)=NSR(IT,26)+NSR(IT,K)
    5 R(IT,26)=R(IT,26)+R(IT,K)
C        THE PERCENT NORMALIZED DIFFERENCES OF THE REGION FRACTIONAL POWERS:
      DO 60  K=1,NK
      DO 50  IT=7,9
   50 R(IT,K)=0.0
      IF (R(6,K).EQ.0.0)  GO TO 60
      R(7,K)=(R(6,K)-R(4,K))*100./R(6,K)
      R(8,K)=(R(6,K)-R(5,K))*100./R(6,K)
      IF (R(5,K).EQ.0.0)  GO TO 60
      R(9,K)=(R(5,K)-R(4,K))*100./R(5,K)
   60 CONTINUE
C        THE PERCENT NORMALIZED DIFFERENCES OF THE TOTAL RAW POWER PRODUCED:
      R(7,26)=(R(3,26)-R(1,26))*100./R(3,26)
      R(8,26)=(R(3,26)-R(2,26))*100./R(3,26)
      R(9,26)=(R(2,26)-R(1,26))*100./R(2,26)
      WRITE (6,10) NK,(K,(NSR(IT,K),R(IT,K),IT=1,3),K=1,NK)
   10 FORMAT ('1RESULTS OF THE INTEGRATED POWER IN EACH OF THE',I3,
     X   ' REGIONS:',///,
     X   '0CALCULATED POWER LEVELS, AND NUMBER OF SUBREGIONS PER REGION:
     X   ',//,3X,'REGION:',10X,'HOMOGENIZED RESULTS:',5X,
     X   'SYNTHESIZED RESULTS:',7X,'REFERENCE RESULTS:',//,
     X   (I10,10X,I3,E17.7,I8,E17.7,I10,E15.7))
      WRITE (6,12) (NSR(IT,26),R(IT,26),IT=1,3)
   12 FORMAT (/,3X,'TOTALS:',10X,I3,E17.7,I8,E17.7,I10,E15.7)
      WRITE (6,20)    (K,(R(IT,K),IT=4,6),K=1,NK)
   20 FORMAT (//,'0FRACTIONAL POWER LEVELS:',//,3X,'REGION:',10X,
```

```
  X    'HOMOGENEOUS RESULTS:',5X,
  X    'SYNTHESIZED RESULTS:',7X,'REFERENCE RESULTS:',//,
  X    (I10,5X,3E25.7))
     WRITE (6,11) (R(IT,26),IT=4,6)
  11 FORMAT (/,3X,'TOTALS:',5X,3E25.7)
     WRITE (6,30)     (K,(R(IT,K),IT=7,9),K=1,NK)
  30 FORMAT (//,'OFRACTIONAL POWER NORMALIZED PERCENT ERRORS:',//,
  X    3X,'REGION:',14X,'(REF-HOMO)/REF %',
  X    8X,'(REF-SYNTH)/REF %',5X,'(SYNTN-HOMO)/SYNTH %',//,
  X    (I10,5X,3E25.7))
     WRITE (6,40) (R(IT,26),IT=7,9)
  40 FORMAT (//,'ORAW PRODUCTION POWER NORMALIZED PERCENT ERRORS:',//,
  X    24X,'(REF-HOMO)/REF %',
  X    8X,'(REF-SYNTH)/REF %',5X,'(SYNTN-HOMO)/SYNTH %',//,
  X    (15X,3E25.7))
     RETURN
     END
```

```
      SUBROUTINE SCPLOT(NK,N,X,U,NB,XB,UB,NC,XC,UC)                     SCPL0001
C     READS IN PLOTTING INFORMATION AND PLOTS THE FLUX COMPARISONS:     SCPL0002
      COMMON /SC/ TITLE(20),XINCH,YINCH,NCELL,WCELL,NLL,XL(100)         SCPL0003
      COMMON /MAX/ XMIN,XMAX,YMIN,YMAX                                  SCPL0004
      COMMON /RASTER/ IXS,IYS,IXE,IYE,IXT,IYT,IXLX,IYLX,                SCPL0005
     X                IXLY,IYLY(3),LH,LW, IS,IR                         SCPL0006
      COMMON /MARGIN/ ML,MR,MB,MT                                       SCPL0007
      DIMENSION  X(1001),U(2,1001),XB(1001),UB(2,1001),XC(1001),        SCPL0008
     X           UC(2,1001)                                             SCPL0009
      READ (5,1,END=20) TITLE                                           SCPL0010
      READ (5,4,END=20) XINCH, YINCH                                    SCPL0011
C        NEGATIVE NCELL SPECIFIES THAT THE LAST CELL IS A HALF CELL:    SCPL0012
      READ (5,3,END=20) NCELL, WCELL                                    SCPL0013
      READ (5,3,END=5) NLL, (XL(I),I=1,7)                               SCPL0014
      IF (NLL.GT.7)  READ (5,4) (XL(I),I=8,NLL)                         SCPL0015
    1 FORMAT (20A4)                                                     SCPL0016
    3 FORMAT (I10,7F10.5)                                               SCPL0017
    4 FORMAT (8F10.5)                                                   SCPL0018
      GO TO 10                                                          SCPL0019
    5 NLL=0                                                             SCPL0020
   10 WRITE (6,11) TITLE                                                SCPL0021
   11 FORMAT ('1EXECUTING GENERAL ANALYSIS AND FLUX PLOTTING PROGRAM:', SCPL0022
     X   //,5X,'TITLE OF PLOTTING RUN IS:    | ',20A4,' |')             SCPL0023
      XMIN=0.0                                                          SCPL0024
      XMAX=WCELL*FLOAT(NCELL)                                           SCPL0025
      IF (NCELL.LT.0)  XMAX=WCELL*(FLOAT(-NCELL)-0.5)                   SCPL0026
      YMIN=0.                                                           SCPL0027
      YMAX=1.0                                                          SCPL0028
      WRITE (6,13) NCELL,WCELL,XMIN,XMAX,YMIN,YMAX,NLL                  SCPL0029
   13 FORMAT ('0REACTOR GEOMETRY PARAMETERS:',/,5X,'NCELL =',I5,/,      SCPL0030
     X   5X,'WCELL =',F10.5,/,5X,'XMIN =',F10.5,/,5X,'XMAX =',F10.5,/,  SCPL0031
     X   5X,'YMIN =',F10.5,/,5X,'YMAX =',F10.5,/,                       SCPL0032
     X   5X,'NLL  =',I5)                                                SCPL0033
      IF (NLL.GT.0) WRITE (6,14)  (XL(I),I=1,NLL)                       SCPL0034
   14 FORMAT (5X,'(XL(I),I=1,NLL) =',/,(10X,10F10.5))                   SCPL0035
      CALL SETUP                                                        SCPL0036
```

```
      CALL PLOT(N,X,U,NB,XB,UB,NC,XC,UC)                    SCPL0037
   20 RETURN                                                SCPL0038
      END                                                   SCPL0039
```

```
      SUBROUTINE SETUP                                                  SETU0001
C     FINDS THE RASTER LOCATIONS FOR THE GRID CORNERS, LABELS, AND TITLE SETU0002
C        ENTIRE PLOTTING AREA (INCLUDING LABELS) IS XINCH X YINCH.      SETU0003
      COMMON /SC/ TITLE(20),XINCH,YINCH,NCELL,WCELL,NLL,XL(100)         SETU0004
      COMMON /RASTER/ IXS,IYS,IXE,IYE,IXT,IYT,IXLX,IYLX,                SETU0005
     X                IXLY,IYLY(3),LH,LW,IS,IR                          SETU0006
      COMMON /MARGIN/ ML,MR,MB,MT                                       SETU0007
C        SET THE LETTER SIZE FOR THE LABELS AND TITLES:                 SETU0008
      LH=2                                                              SETU0009
      LW=2                                                              SETU0010
      IS=5*LW+3                                                         SETU0011
      IR=7*LH+5                                                         SETU0012
C        PLOTTING AREA:                                                 SETU0013
C        ONE INCH = 137 RASTERS. PLOTTING AREA IS 1023 RASTERS SQUARE.  SETU0014
      IXE=XINCH*137.+0.5                                                SETU0015
      IYE=YINCH*137.+0.5                                                SETU0016
      IF (IXE.GT.1020)  IXE=1020                                        SETU0017
      IF (IYE.GT.1020)  IYE=1020                                        SETU0018
      IXS=(1023-IXE)/2.+1                                               SETU0019
      IYS=(1023-IYE)/2+1                                                SETU0020
      IXE=IXS+IXE-1                                                     SETU0021
      IYE=IYS+IYE-1                                                     SETU0022
C        COORDS FOR THE LETTERS ARE FOR THEIR CENTERS:                  SETU0023
C        TITLE (UPPER LEFT CORNER):                                     SETU0024
      IXT=8                                                             SETU0025
      IYT=1014                                                          SETU0026
C        X AXIS LABEL                                                   SETU0027
      IXLX=(IXS+60+IXE-6*IS)/2+1+8                                      SETU0028
      IYLX=IYS + 10                                                     SETU0029
C        Y AXIS LABELS (FOR EACH FLUX PLOT TYPE):                       SETU0030
      IXLY=IXS + 10                                                     SETU0031
      IYLY(1)=(IYS+43+IYE-20*IS)/2+1 +4                                 SETU0032
      IYLY(2)=(IYS+43+IYE-23*IS)/2+1 +4                                 SETU0033
      IYLY(3)=(IYS+43+IYE-26*IS)/2+1 +4                                 SETU0034
C        SET MARGIN SPACING FOR GRID:                                   SETU0035
      ML=IXS + 2*18 -1                                                  SETU0036
```

```
      MR=1023-IXE                                                SETU0037
      MB=IYS + 2*18 -1                                           SETU0038
      MT=1023-IYE                                                SETU0039
      RETURN                                                     SETU0040
      END                                                        SETU0041
```

```
      SUBROUTINE PLOT(NA,XA,UA,NB,XB,UB,NC,XC,UC)
C     USES MIT-IPC'S SC-4020 SUBROUTINE PLOTTING PACKAGE.
C     PLOTS THE COMPARISON FLUX DISTRIBUTIONS:
C        KEY:  HOMOGENEOUS - DASHED LINE.
C              SYNTHESIS - SOLID LINE.
C              REFERENCE - DOTTED LINE.
      COMMON /RASTER/ IXS,IYS,IXE,IYE,IXT,IYT,IXLX,IYLX,
     X                IXLY,IYLY(3)
      DIMENSION  XA(1001),UA(2,1001),XB(1C01),UB(2,1001),
     X           XC(1001),UC(2,1001), X(1001),T(1001)
C        HARDCOPY INITIALIZATION:
      CALL STOIDV('M7788-6571',9,2)
C        DO FOR EACH ENERGY GROUP:
      DO 10  IG=1,2
      NTH=1
      CALL GRID
      IF (IG.EQ.1)  CALL RITE2V(IXLY,IYLY(1),1024,90,1,20,NTH,
     X   'NORMALIZED FAST FLUX',N)
      IF (IG.EQ.2)  CALL RITE2V(IXLY,IYLY(2),1024,90,1,23,NTH,
     X   'NORMALIZED THERMAL FLUX',N)
C        HOMOGENEOUS RESULTS:
      CALL PUT(IG,NC,XC,UC,N,X,T)
      CALL ADJUST(N,X,T,1)
      CALL LINE(N,X,T,1)
C        SYNTHESIS RESULTS:
      CALL PUT(IG,NB,XB,UB,N,X,T)
      CALL ADJUST(N,X,T,0)
      CALL LINE(N,X,T,0)
C        REFERENCE RESULTS:
      CALL PUT(IG,NA,XA,UA,N,X,T)
      CALL ADJUST(N,X,T,2)
      CALL LINE(N,X,T,2)
   10 CONTINUE
      CALL PLTND(NTH)
      WRITE (6,20) NTH
   20 FORMAT ('1',I5,'  SC 4020 PLOTS HAVE BEEN PLOTTED.')
```

```
      RETURN                                                            PLOT0037
      END                                                               PLOT0038
```

```
      SUBROUTINE PUT(IG,NA,XA,UA,N,X,T)
C     TRANSFERS NA,XA,UA OF IG INTC N,X,T:
      DIMENSION  XA(1001),UA(2,1001),X(1001),T(1001)
      N=NA
      DO 1  I=1,N
      X(I)=XA(I)
    1 T(I)=UA(IG,I)
      RETURN
      END
```

```
      SUBROUTINE GRID                                                   GRID0001
C     SETS UP A NEW FRAME AND PLOTS THE GRID                            GRID0002
C     AS WELL AS RUN TITLE, LABLES, AND LIGHT LINES.                    GRID0003
      COMMON /SC/ TITLE(20),XINCH,YINCH,NCELL,WCELL,NLL,XL(100)         GRID0004
      COMMON /MAX/ XMIN, XMAX, YMIN, YMAX                               GRID0005
      COMMON /RASTER/  IXS,IYS,IXE,IYE,IXT,IYT,IXLX,IYLX,               GRID0006
     X                IXLY,IYLY(3),LH,LW,IS,IR                          GRID0007
      COMMON /MARGIN/ ML,MR,MB,MT                                       GRID0008
      DATA IT /1/                                                       GRID0009
      IT=IT+1                                                           GRID0010
      IF (IT.GT.3)  IT=3                                                GRID0011
      NTH=1                                                             GRID0012
      IF (IT.GT.2)  GO TO 10                                            GRID0013
      CALL CHSIZV(LW,LH)                                                GRID0014
      CALL RITSTV(IS,IR)                                                GRID0015
      CALL SETMIV(ML,MR,MB,MT)                                          GRID0016
   10 CALL GRID1V(IT,XMIN,XMAX,YMIN,YMAX,WCELL,0.1,1,0,-1,-1,5,3)       GRID0017
      CALL PRINTV(80,TITLE,IXT,IYT,1)                                   GRID0018
      CALL RITE2V(IXLX,IYLX,1024,0,1,6,NTH,'X (CM)',N)                  GRID0019
   20 IF (NLL.EQ.0)  GO TO 5                                            GRID0020
      IY1=IYV(YMIN)                                                     GRID0021
      IY2=IYV(YMAX)                                                     GRID0022
      DO 1  I=1,NLL                                                     GRID0023
      IX=IXV(XL(I))                                                     GRID0024
    1 CALL LINEV(IX,IY1,IX,IY2)                                         GRID0025
    5 RETURN                                                            GRID0026
      END                                                               GRID0027
```

```
      SUBROUTINE ADJUST(N,X,Y,L)
C     ADJUSTS THE X AND Y ARRAYS BY DELETING ANY (X,Y) POINTS
C     WITHIN OR EQUAL TC NR RASTERS DISTANCE OF EACH OTHER.
C        HOPEFULLY ELIMINATES DARK SPCTS CN THE PLOTS.
C        L = 0:  SOLID LINE PLOTTING:  NR = 2.
C        L = 1:  DASHED LINE PLOTTING: NR = 2;  INCRV(10,5).
C        L = 2:  DOTTED LINE PLOTTING: NR = 4;  INCRV( 2,2).
      DIMENSION X(1), Y(1)
      NR=2
      IF (L.EQ.2)  NR=4
      IF (N.LE.1)  RETURN
      K=1
      IX1=IXV(X(1))
      IY1=IYV(Y(1))
      DO 1  I=2,N
      IX2=IXV(X(I))
      IY2=IYV(Y(I))
      ID=SQRT(FLOAT((IX2-IX1)**2+(IY2-IY1)**2))+0.5
      IF (ID.LE.NR)  GO TO 1
      K=K+1
      X(K)=X(I)
      Y(K)=Y(I)
      IX1=IX2
      IY1=IY2
    1 CONTINUE
      N=K
      RETURN
      END
```

```
      SUBROUTINE LINE(N,X,Y,K)
C     PLOTS:  A LINE (K=0),  OR A DASHED LINE (K=1),  OR A DOTTED LINE (K=2)
C     THROUGH Y(X) DATA POINTS.
      DIMENSION X(1), Y(1)
      IX1=IXV(X(1))
      IY1=IYV(Y(1))
      DO 10  I=2,N
      IX2=IXV(X(I))
      IY2=IYV(Y(I))
      IF (K.NE.0)  GO TO 1
C        SOLID LINE PLOT:
      CALL LINEV(IX1,IY1,IX2,IY2)
      GO TO 5
C        DOTTED OR DASHED LINE PLOT:
    1 IF (K.EQ.1)  CALL INCRV(10,5)
      IF (K.EQ.2)  CALL INCRV(2,2)
      CALL DOTLNV(IX1,IY1,IX2,IY2)
    5 IX1=IX2
   10 IY1=IY2
      RETURN
      END
```