

NUCLEAR ENGINEERING
READING ROOM - M.I.T.

COO-2262-1
MITNE-145

THE FINITE ELEMENT METHOD
APPLIED TO
NEUTRON DIFFUSION PROBLEMS

by

Lothario O. Deppe, K.F. Hansen

February 1973

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

AEC Research and Development Report
Contract AT(11-1)-2262
U.S. Atomic Energy Commission



Room 14-0551
77 Massachusetts Avenue
Cambridge, MA 02139
Ph: 617.253.2800
Email: docs@mit.edu
<http://libraries.mit.edu/docs>

DISCLAIMER

MISSING PAGE(S)

Appendix D has been omitted from the Archives copy. This is the most complete copy available.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF NUCLEAR ENGINEERING

Cambridge, Massachusetts 02139

THE FINITE ELEMENT METHOD
APPLIED TO
NEUTRON DIFFUSION PROBLEMS

by

Lothario O. Deppe, K. F. Hansen

February, 1973

COO - 2262 - 1

MITNE - 145

AEC Research and Development Report

Contract AT(11-1) - 2262

U.S. Atomic Energy Commission

NUCLEAR ENGINEERING
READING ROOM - M.I.T.

THE FINITE ELEMENT METHOD APPLIED TO
NEUTRON DIFFUSION PROBLEMS

by

Lothario O. Deppe

Submitted to the Department of Nuclear Engineering on February 10, 1971, in partial fulfillment of the requirements for the degrees of Master of Science and Nuclear Engineer.

ABSTRACT

The time-independent, multigroup, two-dimensional neutron diffusion equations are solved in the particular case of a piecewise expansion of the unknown flux solutions in bicubic Hermite polynomials. The matrix formulation is achieved by applying the Galerkin method to the "weak" form of the multigroup equations; the matrix problem is solved through the power iteration method and a Cholesky factorization.

Numerical results are presented, as evaluated through a code (CHD), for several small problems and for a 1000 Mw(e) PWR, beginning-of-life core. The main conclusions arrived at through the numerical analysis show the feasibility of extending the domain of the piecewise expansion functions over heterogeneous regions, the actual spatial behavior of the cross sections being accounted for in the inner products arising from the application of the Galerkin method. Results are shown which indicate the accuracy and utility of the particular approach developed in this thesis.

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

TABLE OF CONTENTS

	Page
ABSTRACT	2
LIST OF FIGURES	5
LIST OF TABLES	8
ACKNOWLEDGEMENTS	9
BIOGRAPHICAL NOTE	10
Chapter I - INTRODUCTION	11
1.1 The time-independent neutron diffusion equation and the multigroup approximation	12
1.2 Finite Element Methods	19
Chapter II - A FINITE ELEMENT APPROXIMATION TO THE MULTIGROUP NEUTRON DIFFUSION EQUATIONS	26
2.1 The "weak" form corresponding to the neutron diffusion equations	26
2.2 Construction of a basis for the piecewise expansion in one dimension	32
2.3 The basis functions in two dimensions	38
2.4 Varying cross sections inside the expansion elements	46
2.5 Matrix formulation of the two-group diffusion problem and numerical methods	47
Chapter III - SAMPLE PROBLEMS AND RESULTS	55
3.1 Comparisons between CHD and EXTERMINATOR 2 for a 4-composition problem	56
3.2 The one-dimensional, ten-region problem	63
3.3 The basic two-dimensional, two-composition problem	67
3.4 The basic two-dimensional problem with baffle surrounding the fuel region	72

3.5 The basic two-dimensional problem with varying cross sections inside the fuel region	77
3.6 Results for Zion 1 in two dimensions	84
Chapter IV - CONCLUSIONS AND RECOMMENDATIONS	95
4.1 Conclusions	95
4.2 Recommendations	97
REFERENCES	99
Appendix A - MACROSCOPIC CROSS SECTIONS	101
Appendix B - INNER PRODUCTS BETWEEN FINE LIMITS FOR CUBIC HERMITE POLYNOMIALS	102
Appendix C - THE CODE CHD	107
C.1 General description	107
C.2 Input preparation	114
Appendix D - SOURCE LISTING OF CHD, SAMPLE INPUT AND OUTPUT (Only in MIT library copies)	124

LIST OF FIGURES

	Page
2.1 Function $f_1(x_0)$ and uniquely determined cubic interpolate in region (x_1, x_{i+1})	32
2.2 One-dimensional cubic element functions	34
2.3 Cubic basis functions appropriate for function and derivative continuity	36
2.4 Cubic basis functions appropriate for flux and current continuity across interfaces for a neutron diffusion problem	37
2.5 Linear basis function appropriate for flux continuity across interfaces	38
2.6 Two-dimensional partition	40
2.7 Two-dimensional partition and matrix form for unknowns ordered as: 1) all unknowns at each point; 2) all columns; 3) all rows	51
2.8 Two-dimensional partition and corresponding matrix structure.	52
3.1 The 4-composition problem geometry	56
3.2 Flux plots for 4-composition problem as evaluated through EXTERMINATOR 2 and CHD (cut A)	60
3.3 Flux plots for 4-composition problem as evaluated through EXTERMINATOR 2 and CHD (cut B)	61
3.4 Element averaged results for 4-composition problem as evaluated with EXTERMINATOR 2 and CHD	62
3.4a One-dimensional, ten-region problem	64
3.5 Flux plots for one-dimensional, ten-region problem	66
3.6 The basic two-dimensional, two-composition problem	69

3.7a Thermal flux plots for basic two-dimensional problem at $x=0.0$	70
3.7b Thermal flux plots for basic two-dimensional problem at $x=20.0$ cm	71
3.8 Geometry for the basic two-dimensional problem with baffle surrounding the fuel region	72
3.9a Thermal fluxes for basic two-dimensional problem with baffle surrounding fuel region ($y=0.0$)	74
3.9b Thermal fluxes for basic two-dimensional problem with baffle surrounding fuel region ($y=20.0$)	75
3.10 Geometry for the basic two-dimensional problem with varying cross sections inside the fuel region	77
3.11a Thermal fluxes for two-dimensional problem with varying cross sections inside fuel region ($x=0.0$)	79
3.11b Fast fluxes for two-dimensional problem with varying cross sections inside fuel region ($x=0.0$)	80
3.12a Thermal fluxes for two-dimensional problem with varying cross sections inside fuel region ($x=20.0$)	81
3.12b Fast fluxes for two-dimensional problem with varying cross sections inside fuel region ($x=20.0$)	82
3.13 Zion 1 first core layout	85
3.14 Mesh arrangement for CHD runs (Zion 1)	85
3.15 Zion 1 - Region averaged to core averaged power ratios (finest CHD mesh)	88
3.16 Zion 1 - Region averaged to core averaged power ratios (intermediate CHD mesh)	89
3.17 Zion 1 - Region averaged to core averaged power ratios (coarsest CHD mesh)	90

3.18 Flux plots for Zion 1 as evaluated through PDQ-5 (44 x 44) and CHD (10 x 10)	92
3.19 Flux plots for Zion 1 as evaluated through PDQ-5 (44 x 44), CHD (7 x 7) and CHD (6 x 6)	93
4.1 Regular and virtual mesh lines and points in two dimensions	97
C.1 CHD basic logic	108
C.2 Coordinate system orientation assumed by CHD	109

LIST OF TABLES

	Page
3.1 Overall results for EXTERMINATOR 2 and CHD for the 4-composition problem	58
3.2 Eigenvalues for the one-dimensional, ten-region problem	65
3.3 Element averaged results for the one-dimensional, ten-region problem	65
3.4 Eigenvalues for the basic two-dimensional problem	69
3.6 Overall results for the basic two-dimensional problem with baffle surrounding the fuel region	73
3.7 Overall results for the basic two-dimensional problem with varying cross sections inside the fuel region	78
3.8 Overall results for Zion 1 as evaluated through PDQ-5, CITATION and CHD with several mesh sizes	87
3.9 Relative differences of average power in expansion elements of Zion 1 as evaluated through CHD relative to PDQ-5 and CITATION	91
C.1 Some variable values for CHD runs	111
C.2 Expansion functions option definition	118
C.3 Default sets for expansion functions	120

ACKNOWLEDGEMENTS

The author wishes to express his respect and most grateful appreciation to Professor Kent F. Hansen, his thesis supervisor, for the constant guidance and encouragement provided during the course of this work. Prof. Hansen's accessibility and informality are considered most valuable.

Most of all, this work should be a tribute to the author's wife, Alzira, and to his son, Rodrigo. Encouragement and understanding were always found in them.

The author also wishes to make use of this opportunity to express his respect and gratitude to Mr. Sergio de Salvo Brito, who's enthusiasm and far-reaching views have motivated many Brazilians, the author among them.

Dr. Chang Mu Kang provided valuable guidance in some occasions. The author wishes to thank him for it.

Special thanks are for Furnas Centrais Elétricas S.A. (Rio de Janeiro, Brazil), the authors employer, for providing full support during the author's stay at MIT.

The work was performed under USAEC Contract AT(11-1)-2262, with the computations performed at the MIT Information Processing Center.

Finally, the author wishes to thank Miss Linda Wildman for typing this report with skill and patience.

BIOGRAPHICAL NOTE

The outhor, Lothario Olavo Deppe, was born on September 3, 1940 in Nova Petrópolis, state of Rio Grande do Sul, Brazil. He received elementary school education in his home town and graduated from High School in December, 1959.

In March, 1961 he enrolled at the School of Engineering, University of Rio Grande do Sul, and received his Bachelor of Science Degree, in Electrical Engineering, in December 1965. In 1966 he attended an extension course in Nuclear Engineering at the Military Institute of Engineering in Rio de Janeiro.

In 1967 he joined the Brazilian Nuclear Energy Commission where he was mostly connected with economic studies regarding the introduction of nuclear plants in brazilian power systems. In 1969 he worked briefly as an independent consultant to Instituto de Pesquisas Radioativas from Minas Gerais, Brazil, and in May of this year he joined the staff of Furnas Centrais Eletricas S.A. from Rio de Janeiro, where he has recently been appointed as head of the nuclear section in the Planning Department.

He enrolled at the Massachusetts Institute of Technology as a candidate for the degree of Nuclear Engineer in September, 1970.

The author is married to the former Alzira Soares Lourenço and has a son.

INTRODUCTION

Nuclear reactor physics problems have an inherent numerical connotation, in the sense that they are so complex that no practical problem can be solved by analytic solution of the relevant equations. Consequently, approximation methods are of paramount importance in solving reactor physics problems. Conversely, given some convenient numerical procedure to find an approximate solution to an equation, the accuracy of the solution is strongly dependent upon the computer capacity available to carry out the computations. As a consequence, two ways are open to improve the accuracy with which the behavior of a nuclear reactor can be predicted: better computer performance and inherently more accurate numerical methods.

The present thesis deals with the second alternative and tries to demonstrate that the Finite Element Method has a potential to improve, at least in one particular form, the accuracy with which present, practical, time-independent neutron diffusion equations can be solved.

Neutron diffusion equations are characterized by heterogeneities in the regions in which solutions are sought. If the maximum mesh size of the finite element method were to be limited by the size of the heterogeneity, we would have a natural minimum to the number of unknowns of the problem, and would not be able to take full advantage of the method's inherent high accuracy. In addition it is to be expected that finite element methods will ultimately compare advantageously with finite difference methods if they

result in a considerably lower number of unknowns. Therefore, one would like to have the possibility of defining larger meshes than the heterogeneities themselves (for a PWR in two dimensions, for instance, over 4, 9 or even 16 fuel elements). This possibility is investigated in this thesis for practical problems.

In the remainder of this Chapter we will present the basic, time-independent, continuous energy, homogeneous neutron diffusion equation; we will discuss its meaning and present some of the methods most often used in its solution; finally we will introduce the Finite Element Method itself.

In Chapter II we will discuss the mathematical preliminaries leading to the final matrix equations that have to be solved, and the numerical methods used to achieve this.

In Chapter III we present a series of actual results that were obtained by the use of the computer code CHD, written in order to implement the method in two dimensions and to test the use of coarse meshes on realistic systems. Results are obtained for numerous test problems; in particular we examine in some detail a representation of the Zion 1 reactor in two dimensions, and for three different coarse meshes: one fuel element, 4 fuel elements in some locations, and up to 16 fuel elements in the innermost region.

1.1) The time-independent neutron diffusion equation and the multigroup approximation

We define a closed energy domain $\xi = [E_{\min}, E_{\max}]$, where E_{\min} and E_{\max} are conveniently chosen, in terms of the physical characteristics of the system, as lower and upper bounds of the neutron energy.

In usual reactor problems the mathematical model to the physical problem is defined so that all the problem parameters are at least piecewise continuous, discontinuities being allowed across the interfaces. Therefore, we define the open volume domain Ω , which is the union of a finite number (L) of contiguous open subdomains Ω_1 .

$$\Omega = \bigcup_{1=1}^L \Omega_1$$

The collection of interfaces is denoted $\partial_1 \Omega$, and the boundary $\partial \Omega$ is the union of the exterior boundary and the interfaces

$$\partial \Omega = \partial_e \Omega \cup \partial_1 \Omega$$

In each Ω_1 the properties are assumed to be continuous with discontinuities allowed at $\partial_1 \Omega$.

We also define the open domain $\bar{\Omega}$

$$\bar{\Omega} = \Omega \cup \partial_1 \Omega$$

The continuous energy, homogeneous, time-independent neutron diffusion equation can be written as (ref. 1):

$$\begin{aligned} & -\underline{\nabla} \cdot D(\underline{r}, E) \underline{\nabla} \phi(\underline{r}, E) + \Sigma_t(\underline{r}, E) \phi(\underline{r}, E) \\ = & \int_{\xi} dE' \Sigma_s(\underline{r}, E' \rightarrow E) \phi(\underline{r}, E') + \frac{\chi(E)}{\lambda} \int_{\xi} dE' \nu \Sigma_f(\underline{r}, E') \phi(\underline{r}, E') \quad (1.1.a) \end{aligned}$$

We will impose any of the following homogeneous boundary conditions:

$$\phi(\underline{r}, E) = 0$$

$$\text{or } \frac{\partial}{\partial \underline{n}} \phi(\underline{r}, E) = 0$$

$$\begin{aligned} &\text{for } E \in \xi \\ &\text{and } \underline{r} \in \partial_e \Omega \end{aligned}$$

(1.1.b)

In addition, for all interface points we require flux and current continuity, i.e.

$$\phi(\underline{r}^-, E) = \phi(\underline{r}^+, E)$$

$$D(\underline{r}^-, E) \frac{\partial}{\partial \underline{n}} \phi(\underline{r}^-, E) = D(\underline{r}^+, E) \frac{\partial}{\partial \underline{n}} \phi(\underline{r}^+, E)$$

$$\begin{aligned} &\text{for } E \in \xi \\ &\text{and } \underline{r} \in \partial_i \Omega \end{aligned}$$

(1.1.c)

The symbols have the following meaning:

\underline{r} - generalized spatial variable

E - energy variable

$\phi(\underline{r}, E)$ - scalar neutron flux (/cm².s)

$D(\underline{r}, E)$ - diffusion coefficient (cm)

$\Sigma_t(\underline{r}, E)$ - macroscopic total cross section (/cm)

$\Sigma_s(\underline{r}, E' \rightarrow E)$ - macroscopic scattering cross section from E' to E (/cm)

$\chi(E)$ - prompt fission neutron spectrum

$\nu \Sigma_f(\underline{r}, E)$ - fission neutron production macroscopic cross section (/cm)

$\lambda = 1/k_{\text{eff}}$ - eigenvalue introduced to define an eigenvalue problem

k_{eff} - system multiplication factor

$\partial/\partial \underline{n}$ - outward normal derivative at a surface.

Eq. (1.1.a) results from a balance condition applied to a volume $d\mathbf{r}$ about \mathbf{r} and to an energy interval dE about E . The left-hand side specifies losses (or gains) by net leakage and by removal; the right-hand side specifies births by inscattering from other energies or by fissions in the whole energy domain.

To simplify notation, Eq. (1.1) will henceforth be expressed as

$$L \phi = 0$$

where L is a linear operator.

The exact solution of the boundary value problem (1.1) is impossible for any real life case. To begin with, there exists no analytic function representing the detailed energy dependence of any of the cross sections throughout the energy domain. This complexity leads to the necessity of a multigroup formalism for the energy variable: one partitions the energy domain into a finite set $(\epsilon_1, \dots, \epsilon_G)$ of contiguous subdomains and builds up a coupled set of differential equations resulting from the balance condition applied to each subdomain, or energy group.

$$A_1 \phi_1 = -\nabla \cdot D_1 \nabla \phi_1 + \Sigma_{r1} \phi_1 = \frac{1}{\lambda} \sum_{g=1}^G v \Sigma_{fg} \phi_g$$

$$A_2 \phi_2 = -\nabla \cdot D_2 \nabla \phi_2 + \Sigma_{r2} \phi_2 = \Sigma_{s1} \phi_1$$

$$A_G \phi_G = -\nabla \cdot D_G \nabla \phi_G + \Sigma_{rG} \phi_G = \Sigma_{s(G-1)} \phi_{G-1} \quad (1.2)$$

Where the spatial variable \mathbf{r} is implied in the notation, and the groups are numbered by beginning with the highest energies. The A_g are linear, self-adjoint operators.

The group solutions ϕ_g in Eq. (1.2) are all required to satisfy the continuity and boundary conditions (1.1.b) and (1.1.c).

The lower indices used in (1.2) specify energy group. Otherwise the symbols have identical meanings as in (1.1), except for Σ_{rg} which means the total removal cross section from group g and Σ_{sg} which means the outscattering cross sections for group g .

In building up Eq. (1.2) we have assumed (and we will do so henceforth) that all fission events give birth to neutrons in group 1 and that outscattering from any group (g) leads to group ($g+1$). None of these assumptions are essential, in any sense, to the application of the finite element method to Eq.(1.2). However, they simplify calculations and data handling. The first one arises frequently from the few-group structure with which multi-dimensional diffusion calculations are usually made for thermal reactors; the second one is very accurate in most practical cases and is used in PDQ-5 (ref.2) and PDQ-7 (ref.3), for instance.

It is assumed also that it has been possible to do a reasonable homogenization by auxiliary means in order to find the group parameters.

The most important method existing for the solution of Eq. (1.2) is based on the finite difference approximation to the relevant differential operators with the coupling coefficients generated by the "box integration" method (ref 4). This technique is used in the PDQ codes with few groups and yields low order accuracy. Consequently a very fine mesh has to be imposed on the system if reasonably accurate results are to be obtained. The resultant system of simultaneous equations is very large so that not all data can be held in the fast memory of the computers and sophisticated

iterative techniques have to be used in the solution.

Nodal methods have been developed for cases when speed of execution rather than accuracy is the relevant factor. The FLARE code (ref.5) is a good example of this approach: it is essentially a one-group, three-dimensional model, using a very coarse mesh, and thus requiring few unknowns. The loose coupling between the coarse nodes requires that FLARE be "normalized" to more accurate codes for each type of problem; this procedure introduces an "ad hoc" component and is a serious drawback of the method. However, the code is very fast running and satisfactory for some calculations.

Synthesis methods are now being tried for the solution of detailed, three-dimensional problems. In this approach one tries to break the solution of a three-dimensional problem into two pieces. First, a series of two-dimensional solutions is obtained for typical cross sections of the reactor in the third dimension. Next, the overall, three-dimensional solution is assumed to be a linear combination of the two-dimensional solutions, and one "hopes" the procedure to be accurate.

The drawback of the synthesis methods is the lack of error bounds on the solution: one cannot be sure to improve the results by decreasing the mesh. In the finite difference methods, on the other hand, one can be sure to improve the solution by decreasing the mesh, although the convergence rate is very low. Finite element methods also show this nice property, the convergence rate being usually much higher than for the finite difference method.

Methods based on Monte Carlo simulations have also been used in reactor analysis, and have sometimes drawn great enthusiasm as being the ultimate method. It is based on the idea of following "histories" of neutrons, with the several possible events chosen in a random way. The accuracy which can be achieved is thus dependent on the number of "histories" that can be followed, and this number may be very high if a very fine detail is desired. Even on modern computers, such a calculation may become too costly, and it is therefore not likely that Monte Carlo methods will ultimately displace all other methods.

It seems clear that there still is room for improvement in reactor analysis. There exist many codes available and for numerous calculations the speed and precision attained is acceptable. However, there exist also areas where the present state is not satisfactory: three-dimensional, fine mesh, multigroup, static calculations are not yet feasible in modern computers, and three-dimensional, time-dependent problems have barely been touched. On the other hand, the increased concern with reactor safety today, and the increased cost of the incorporation of safety margins in designs, will undoubtedly require the prediction of performances with ever higher accuracy, if possible at low cost.

The solution of the static diffusion equation is probably the single most important calculation in the whole field known as reactor analysis, because it is an inherent part of depletion and kinetics methods, because it is intrinsically three-dimensional, and because it has to be done so often. In this thesis we will show that finite element methods are feasible in some real-life, static, two-dimensional diffusion problems and, furthermore, that they show a potential for the development of faster codes.

1.2) Finite element methods

Finite element methods have been successfully applied in many practical problems where natural discontinuities exist in the domain where solutions are to be found. The general idea behind the method is very simple: given the equation of a problem, one seeks to find approximate solutions which are defined only over subdomains (or finite elements) of the problem domain. When put together, these piecewise solutions form an approximate solution over the whole problem domain. Methods exist by which one can provide some continuity over the interfaces of the elements; this requirement may not be a necessity, although it is often used because it results in the reduction of the number of unknowns.

One central problem consists, then, in finding appropriate, piecewise continuous functions to use as trial functions in an appropriate space, where the approximate solutions will be sought. Polynomials are a natural choice because they are easily integrated and differentiated, they are familiar, and can be made complete. Other trial functions (such as sines and cosines) can be used with relative ease and it is not clear yet whether their use may improve the method for neutron diffusion problems.

Among the possible polynomial spaces, those based upon Lagrange and Hermite methods have been applied to diffusion problems: the first ones are generated by using function values at internal points, whereas Hermite polynomials are generated by values of the function and derivatives at the boundary of the subdomains. A desirable property of Hermite polynomials is the fact that derivatives and/or function continuities can be naturally imposed on the trial functions, depending on our desire to do so and on the degree of the polynomials.

The reduction from the continuous to the discrete form is generally achieved by the application of the Galerkin method to the "weak" form of the equation. The resultant matrix equation appears as

$$\underline{A} \underline{x} = \frac{\underline{F}}{\lambda} \underline{x}$$

where \underline{x} is the unknown vector solution, λ an eigenvalue, and \underline{A} and \underline{F} are the coefficient matrices.

Sparse coefficient matrices are a general consequence of the application of the method, because the piecewise character of the expansion functions assures that only neighboring points couple together. This is a desirable property because it makes the matrices relatively well-conditioned for inversions, and reduces the number of computer operations.

In some problems there exist natural physical regions where it is to be expected that the solution be continuous and relatively smooth within the region. When this is the case one will probably choose such natural regions as the expansion elements. Nuclear reactors, in particular, present such features: in the beginning-of-life case of a PWR, for instance, natural expansion elements are the fuel elements themselves, which are taken as having constant cross sections. As the reactor is depleted, flux shapes tend to get smoother but one may still expect to find discontinuities of cross sections across the interfaces of the fuel elements. These facts and the need, which was discussed earlier, for more accurate and less expensive ways of solving the neutron diffusion equation, makes the application of the finite element method look attractive

beforehand. The amount of work that has been carried out to date in this area is not as large as, for instance, has been done in the area of structural engineering. Several questions remain unclear, and experiences are not completely conclusive yet.

Ref. 6 and ref. 7 present a very good discussion of the method when applied to static and time-dependent neutron diffusion problems, and to slowing down problems in infinite homogeneous media, using Hermite piecewise expansion. Numerical examples are presented for the one and two-dimensional static diffusion equations, as well as for the zero, one and two-dimensional kinetics equations.

A rectangular partition of the space domain is always assumed, and the cross sections are not allowed to vary inside the elements. The relevant mathematical implications of the method are very well analysed. Trial functions are presented which naturally preserve flux and current continuity in neutron diffusion problems, whereas practical ways of accounting for singularities are tested.

A Lagrangian approximation for a triangular partition is presented in ref. 8. Results are shown for four problems in two energy groups, including a 5-region LMFBR configuration. The results are highly encouraging both in execution time and in the accuracy achieved in the prediction of k_{eff} .

A Lagrangian approximation with linear functions is also considered in ref. 9, both for a mixed rectangular-triangular partition and for a pure rectangular partition. The desirability of the mixed partition is demonstrated.

Triangular partitions are in general very flexible in locating expansion elements. The introduction of a new triangle at a region where more detail is desired will not automatically require the introduction of nodes at regions where they are not desired. In addition, curved geometries are easily followed.

Other ideas have been tried elsewhere; e.g. in ref. 10 results are presented for "modulated" piecewise polynomials. Detailed element solutions, obtained by imposing zero derivative conditions on the boundaries of a heterogeneous region, are modified by piecewise Hermite polynomials and used directly as trial functions in coarse mesh, one-dimensional, two-group calculations. The coarse mesh is defined over the whole heterogeneous region for which the detailed solution was obtained, so that the method does actually yield great flux shape details with coarse meshes. No error bounds were found for this scheme but the numerical results were encouraging.

Except for ref. 10, nowhere has there been made any attempt to allow for the variation of parameters inside the expansion elements so as to take full advantage of the finite element method's inherent high accuracy. In the present thesis this interesting possibility will be investigated in a general way. We will see that when the approximation to the "weak" form of the diffusion equation is sought, the spatial variation of the parameters can be taken into account very naturally; furthermore, we will see that numerous numerical experiments indicate that the scheme is worthwhile pursuing.

We will next outline in very general terms the application of the finite element method in order to set up the general structure of Chapter II. Given the equation of a problem, it involves usually a 4 - step procedure.

a) Define a convenient partition of the problem domain, on a bona fide basis, taking into consideration the characteristics of the system, so as to arrive at expansion elements with reasonably constant properties. This choice is frequently natural, as in the case of neutron diffusion problems.

Define, consequently, the open expansion elements Ω_1 , the collection of interfaces $\partial_1 \Omega$ and the exterior boundary $\partial_e \Omega$.

b) Choose convenient trial functions for the expansion inside the elements. At this stage considerable freedom exists and the choice will depend on what the user expects to be the general behavior of the solution inside the elements, and on the desirability (or not) of some continuity across the interfaces. Cubic Hermite polynomials, for instance, should be chosen when one expects to be able to preserve function and first derivative continuity. A Lagrangian approximation will only preserve function continuity.

Error bounds of the approximation can be found at this stage.

c) Select a procedure to set up the algebraic set of coupled equations which will permit the solution of the system. The Galerkin method has been widely and successfully used, in general in connection with the "weak" formulation which we will proceed to introduce.

Let the model problem, for instance, be the following:

$$-\nabla^2 \phi = L\phi = f \quad (1.3)$$

where L is a linear operator over the region $\bar{\Omega}$. The usual continuity and boundary conditions are imposed: function and first derivative continuity across interfaces and homogeneous boundary conditions.

The boundary problem can be formulated in a variational form. The functional

$$J = (\phi, L\phi) - 2(\phi, f)$$

is stationary provided

$$(u, L\phi) - (u, f) = 0 \quad (1.4)$$

for all u in the same space as ϕ itself. The $(,)$ indicate the inner products.

An integration by parts yields

$$(\nabla\phi, \nabla u) - \int_S u \frac{\partial \phi}{\partial \underline{n}} ds - (u, f) = 0$$

Since homogeneous boundary and derivative continuity conditions have been imposed we see that the surface integral vanishes and we are left with

$$(\nabla\phi, \nabla u) - (u, f) = 0 \quad (1.5)$$

Eq. (1.5) is called the "weak" form of Eq. (1.4).

We next seek an approximate solution, in the convenient space of piecewise functions $u_i(\underline{r})$, to Eq. (1.3).

$$\hat{\phi} = \sum_{i=1}^N a_i u_i(\underline{r}) \quad (1.6)$$

The Galerkin method would require that

$$(u_i, L\phi) = (u_i, f) \quad i = 1, 2, \dots, N$$

There are serious drawbacks in this scheme. The expansion functions are required to be twice differentiable and the interface conditions are imposed on the approximate solution. Thus, in order to increase our degree of freedom in choosing the expansion functions we seek instead an approximate solution to the "weak" form Eq. (1.5). The Galerkin method then requires

$$(\nabla u_i, \nabla \phi) = (u_i, f) \quad i = 1, 2, \dots, N$$

(1.7)

The crucial fact in this step is that the space of trial functions is enlarged. Linear functions, for instance, are allowed in the "weak" form, and the approximate solution is not required to obey the derivative continuity conditions.

d) The fourth step in the application of the finite element method is the solution of the set of coupled equations (1.7) and the regeneration of the approximate solution through Eq. (1.6).

CHAPTER II

A FINITE ELEMENT APPROXIMATIONTO THE MULTIGROUP NEUTRON DIFFUSION EQUATIONS

In this chapter we will apply piecewise Hermite polynomial approximations to the multigroup diffusion equations. We first show how the Galerkin method is applied when the multigroup solutions are expanded in terms of piecewise trial functions; we further show the derivation of the "weak" forms corresponding to the equations, and discuss its implications and advantages.

Next we shall show the development of the convenient basis functions for the Hermite piecewise approximation of the multigroup neutron diffusion equations. We then discuss methods to preserve continuities of the solution and "ad hoc" ways to take into account the singularities.

We also discuss the application of the method to cases when the cross sections are allowed to vary inside the expansion elements, and show what the implications of this fact are.

Finally we show the matrix formulation of the problem and the numerical methods used for the solution of the system in the computer code CHD written to implement the method in two-dimensional, multigroup diffusion problems.

2.1) The "weak" form corresponding to the neutron diffusion equations

In this section we develop the coupled "weak" form of the multigroup diffusion equations, arrived at when Eqs. (1.2) are

multiplied by any allowed weighting functions and integrated over the problem domain. We then show the inherent advantages of seeking approximate solutions to the "weak" form instead of to the diffusion equations themselves.

We first draw our attention back to Eqs. (1.2), the basic multigroup diffusion equations, which are repeated here together with the boundary and continuity conditions.

$$\begin{aligned} A_1 \phi_1 &= \frac{1}{\lambda} \sum_{g=1}^G v \Sigma_{fg} \phi_g \\ A_g \phi_g &= \Sigma_{sg-1} \phi_{g-1} \quad g = 2, 3, \dots, G \end{aligned} \quad (1.2)$$

with boundary conditions

$$\begin{aligned} \phi_g &= 0 \\ \frac{\partial \phi_g}{\partial \underline{n}} &= 0 \quad g = 1, 2, \dots, G \\ & \quad r \in \partial_e \Omega \end{aligned}$$

and continuity conditions

$$\begin{aligned} \phi_g^- &= \phi_g^+ \\ D_g^- \frac{\partial \phi_g^-}{\partial \underline{n}} &= D_g^+ \frac{\partial \phi_g^+}{\partial \underline{n}} \quad g = 1, 2, \dots, G \\ & \quad r \in \partial_1 \Omega \end{aligned}$$

The spatial variable is implied in the notation.

In a more compact matrix notation the multigroup diffusion equations can be written as

$$\underline{A} \underline{\phi}(\underline{r}) = \frac{1}{\lambda} \underline{F} \underline{\phi}(\underline{r}) \quad (2.1)$$

where $\underline{\phi}$ is the group flux vector

$$\underline{\phi} = [\phi_1 \ \phi_2 \ \dots \ \phi_G]^T$$

where \underline{A} and \underline{F} are $G \times G$ matrices, and ϕ is required to satisfy the same continuity and boundary conditions as Eq. (1.2).

It can be shown (ref. 7) that the multigroup "weak" form corresponding to the multigroup equations (1.2) can be derived directly from the continuous energy diffusion equation (1.1) by defining convenient weighting functions. In this thesis we will not use this rigorous procedure, because it is not a crucial step. Instead, we will assume the multigroup equations as arrived at independently and we will seek approximations to the group solutions themselves. In using this approach we will be unable to present any measure of the error bounds on the solution in the energy variable, so that in Sections 2.2 and 2.3, where error bounds will be presented for the Hermite approximation, we will be concerned only with the spatial variables.

In particular, we will not be able to evaluate error bounds on the eigenvalue of the system. However, the multigroup formalism is a well-established approximation and there exist many homogenization procedures which have been shown to present sufficiently accurate results. The main problem in the solution of a diffusion equation lies in the spatial approximation, and it is with this problem that this thesis is concerned with.

We next seek approximate solutions to the G coupled equations implied in Eq. (2.1) by an expansion in some conveniently chosen $N \times G$ piecewise functions, continuous in each Ω_1 . The expansion coefficients a_1^g are not to be confused with the elements of the matrix \underline{A} , which are never used in the following discussion.

$$\underline{\hat{\phi}}(\underline{r}) = \sum_{i=1}^N \underline{U}_i(\underline{r}) \underline{a}_i = [\hat{\phi}_1(\underline{r}) \dots \hat{\phi}_N(\underline{r})]^T \quad (2.2)$$

where

$$\underline{a}_i = [a_i^1 \dots a_i^G]^T \quad i = 1, 2, \dots, N$$

are the unknown coefficient vectors and

$$\underline{U}_i = \text{Diag} [u_i^1(\underline{r}) \dots u_i^G(\underline{r})] \quad i = 1, 2, \dots, N$$

are the expansion matrices.

We have seen earlier (Ch.I) that we would be very limited in the choice of the expansion function if we were to seek approximate solutions directly to Eq. (2.1), because of the interface conditions implied in this equation. The use of the Galerkin method would require that the introduction of Eq. (2.2) into Eq. (2.1) result in

$$(\underline{U}_i, \underline{A} \underline{\hat{\phi}} - \frac{1}{\lambda} \underline{F} \underline{\hat{\phi}}) = 0 \quad i = 1, 2, \dots, N \quad (2.3)$$

where (,) means the inner product.

We will now focus attention on what would be required for the terms of (2.3) to exist. The nature of the operator \underline{A} is such that we must require that the $D \underline{\nabla} \underline{\hat{\phi}}$ be continuous (which is, in fact, the current continuity condition) so that the $(u_i, \underline{\nabla} \cdot D \underline{\nabla} \underline{\hat{\phi}})$ exist. We will have great difficulties in finding expansion functions which will provide this continuity: piecewise linear functions are excluded at once; furthermore, we will see that in the vicinity of singular points we do not, in fact, want to preserve current continuity.

These facts lead to the desirability of seeking solutions in the "weak" form of Eq. (2.1) where the continuity conditions can be relaxed. To achieve this we assume $\underline{\phi}$ to be a solution to Eq. (2.1). Then, for any compatible diagonal matrix \underline{W} we must have

$$(\underline{W}, \underline{A} \underline{\phi} - \frac{1}{\lambda} \underline{F} \underline{\phi}) = 0$$

Integrate the leakage term by parts to obtain

$$(\underline{\nabla} \underline{W}, \underline{D} \underline{\nabla} \underline{\phi}) + (\underline{W}, \underline{\Sigma}_R \underline{\phi}) + \int_S ds \underline{W} \underline{D} \underline{\nabla} \underline{\phi} = (\underline{W}, \frac{1}{\lambda} \underline{F} \underline{\phi})$$

Because of the continuity and boundary conditions the integrals vanish and we are left with a bilinear form

$$a(\underline{W}, \underline{\phi}) = (\underline{\nabla} \underline{W}, \underline{D} \underline{\nabla} \underline{\phi}) + (\underline{W}, \underline{\Sigma}_R \underline{\phi}) - (\underline{W}, \frac{1}{\lambda} \underline{F} \underline{\phi}) = 0 \quad (2.4)$$

where

$$\underline{D} = [D_1 \dots D_G]^T$$

$$\underline{\Sigma}_R = [\Sigma_{r1} \dots \Sigma_{rG}]^T$$

In order for the terms of the bilinear form (2.4) to exist we must only require that $\underline{\phi}$ and \underline{W} and their first spatial derivatives be squared integrable. The space of functions which satisfy these conditions is called a Sobolev space.

In order that $\hat{\phi}$ be an approximate solution to the "weak" form Eq. (2.4), the Galerkin method requires that

$$a(\underline{U}_1, \hat{\phi}) = 0 \quad i = 1, 2, \dots, N \quad (2.5)$$

This scheme thus does enlarge considerably the number of piecewise functions u_i that can be used in the expansion: in particular, linear piecewise functions can be used and the interface current continuity conditions do not have necessarily to be met. We will only require function continuity.

In the next two sections we will summarize the construction of a complete basis for the piecewise expansion of the solution to the "weak" form of the multigroup neutron diffusion equations, in one and two dimensions, and in terms of the Hermite polynomials of the first and third degrees.

Ref. 6 and Ref. 7 discuss in great detail the construction of such bases. We will present it in a more intuitive way, summarize the highlights and present final conclusions.

In neutron diffusion problems we are generally faced with regions whose properties are piecewise smooth. For a beginning-of-life core of a PWR, for instance, the fuel elements are generally considered as homogenized mixtures of the actual elements, the mixture being defined with constant nuclear properties throughout the element. Once the reactor is depleted, the properties will change in a non-uniform way inside the elements. However, they will still be relatively smooth and we probably will have to consider discontinuities at interfaces.

Furthermore, we have imposed flux and current continuity conditions to Eq. (2.1). These conditions are natural for neutron diffusion equations and this fact, when taken together with the piecewise smoothness of the nuclear properties leads naturally to the use of the Hermite interpolation. In this scheme, interpolating polynomials (called Hermite polynomials) are generated inside the interpolation domain by the use of function values and derivatives at the boundaries.

2.2) Construction of a basis for the piecewise expansion in one dimension

We will now focus attention on the construction of the Hermite cubic basis appropriate for the expansion of the unknown solution in a one-dimensional space and indicate afterwards the use of linear bases and, in the next section, the extension to two-dimensions. Thus consider a one-dimensional interval (x_i, x_{i+1}) as shown in Fig. (2.1), part of a region (a,b) , where we want to interpolate an unknown function, knowing the values of the function and of the first derivatives at both ends. It is easily seen that the cubic polynomial is uniquely determined by these values inside the region of interest.

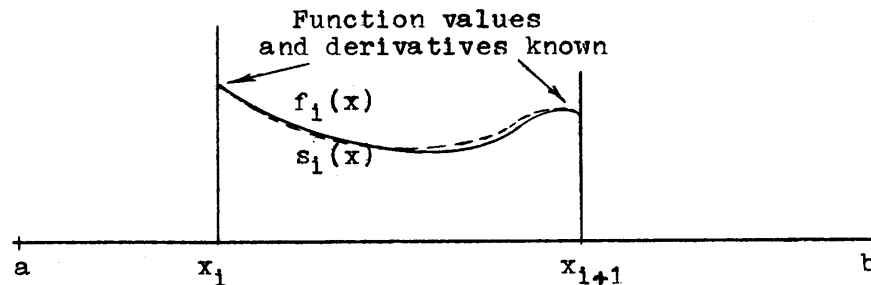


Fig.2.1 - Function $f_1(x)$ and uniquely determined cubic interpolate in region (x_i, x_{i+1}) .

The pointwise error bound of the general Hermite interpolation is well known and stated in ref. 6 and 7. In the maximum norm inside (x_i, x_{i+1}) and for one-dimensional neutron diffusion problems the interpolation error is bounded by:

$$\left\| \frac{d^q}{dx^q} [f_i(x) - s_i(x)] \right\| \leq K |x_{i+1} - x_i|^{(\mu-q)}$$

$$0 \leq q \leq m-1$$

Where μ in our case is

$$\mu = 2m$$

K is a positive constant independent of $|x_{i+1} - x_i|$, and m is such that $(2m-1)$ is the degree of the interpolating polynomial.

We further realize that $s_i(x)$ can be expressed in the following way, by a sum of four polynomials:

$$s_i(x) = f(x_i) u_i^{0+}(x) + f(x_{i+1}) u_{i+1}^{0-}(x)$$

$$+ f(x_i) u_i^{1+}(x) + f(x_{i+1}) u_{i+1}^{1-}(x)$$

where the primes denote x -differentiation.

Each of the u 's is uniquely defined; they are called element functions in refs. 6 and 7, form a complete piecewise basis inside (x_i, x_{i+1}) , and are expressed by

$$u_{i+1}^{0-}(x) = \begin{cases} 3\left[\frac{x-x_1}{x_{i+1}-x_1}\right]^2 - 2\left[\frac{x-x_1}{x_{i+1}-x_1}\right]^3 & , x_1 \leq x \leq x_{i+1} \\ 0 & , \text{Otherwise} \end{cases} \quad (2.6.a)$$

$$u_i^{0+}(x) = \begin{cases} 3\left[\frac{x_{i+1}-x}{x_{i+1}-x_1}\right]^2 - 2\left[\frac{x_{i+1}-x}{x_{i+1}-x_1}\right]^3 & , x_1 \leq x \leq x_{i+1} \\ 0 & , \text{Otherwise} \end{cases} \quad (2.6.b)$$

$$u_{i+1}^{1-}(x) = \begin{cases} \left[-\left(\frac{x-x_1}{x_{i+1}-x_1}\right)^2 + \left(\frac{x-x_1}{x_{i+1}-x_1}\right)^3 \right] (x_{i+1}-x_1) & , x_1 \leq x \leq x_{i+1} \\ 0 & , \text{Otherwise} \end{cases} \quad (2.6.c)$$

$$u_i^{1+}(x) = \begin{cases} \left[\left(\frac{x_{i+1}-x}{x_{i+1}-x_1}\right)^2 - \left(\frac{x_{i+1}-x}{x_{i+1}-x_1}\right)^3 \right] (x_{i+1}-x_1) & , x_1 \leq x \leq x_{i+1} \\ 0 & , \text{Otherwise} \end{cases} \quad (2.6.c)$$

In Fig. 2.2 we plot these functions.

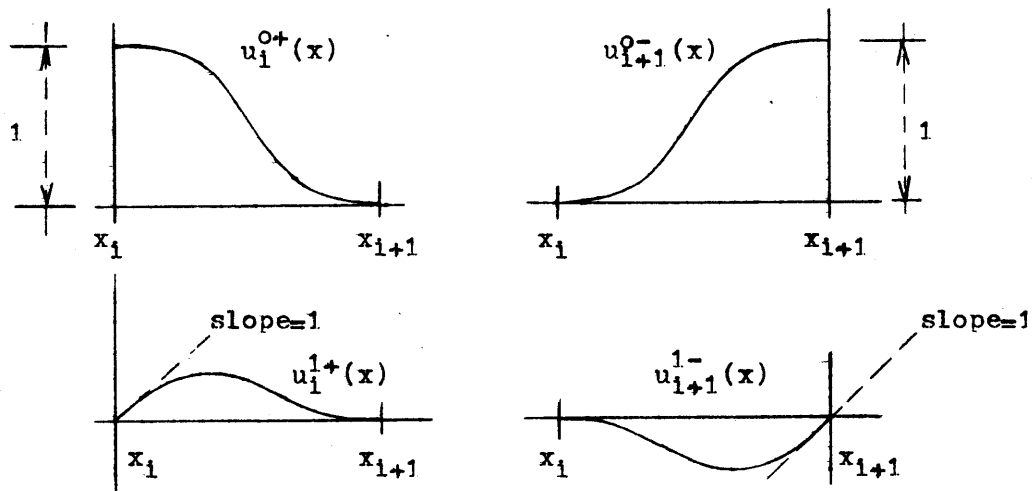


Fig.2.2 - One-dimensional cubic element functions

If, now, the region (a,b) is considered as partitioned in (N-1) contiguous sub-domains, we can approximate the unknown solution of a problem over the whole interval by a sum of piecewise approximations with the use of a set of u-functions in each sub-domain.

$$\begin{aligned}
 s(x) &= \sum_{i=1}^{n-1} s_i(x) \\
 &= \sum_{i=1}^{n-1} [f(x_i^+) u_i^{0+}(x) + f(x_{i+1}^-) u_{i+1}^{0-}(x) \\
 &\quad + f'(x_i^+) u_i^{1+}(x) + f'(x_{i+1}^-) u_{i+1}^{1-}(x)]
 \end{aligned}$$

where the f's are unknowns to be determined.

Function and derivative continuity are naturally preserved by making

$$\begin{aligned}
 f(x_i^+) &= f(x_i^-) = f(x_i) \\
 f'(x_i^+) &= f'(x_i^-) = f'(x_i) \\
 i &= 2, N-1 \\
 &\quad (2.7)
 \end{aligned}$$

Conditions (2.7) lead to the generation of the basis functions appropriate for the approximation of an unknown solution inside the interval (a,b), preserving function and derivative continuity across the interfaces. They are shown in Fig. 2.3

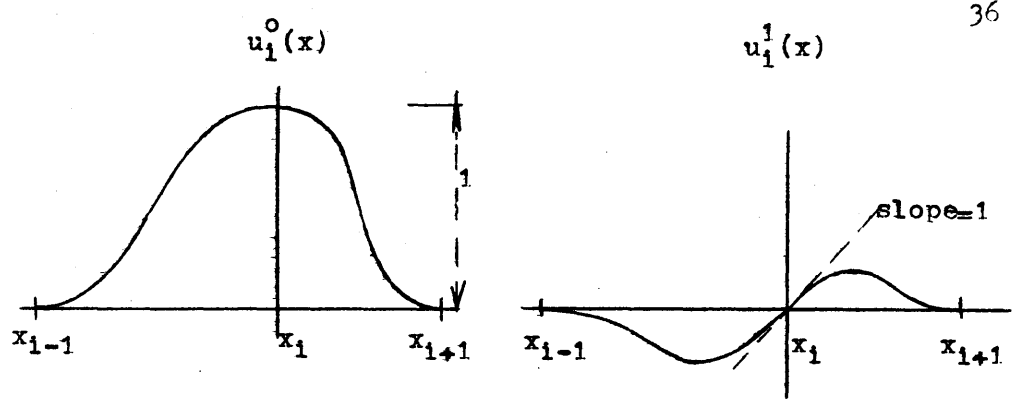


Fig.2.3 - Cubic basis functions appropriate for function and derivative continuity

In neutron diffusion problems, however, we want to require current continuity across the partition interfaces. This condition can be readily met if we redefine slightly the 1-type element functions in the following way, in two contiguous subdomains (see Fig. 2.4).

$$u_i^1(x) = \begin{cases} (\frac{\theta}{D^-}) u_i^{1-}(x), & x_{i-1} \leq x \leq x_i \\ (\frac{\theta}{D^+}) u_i^{1+}(x), & x_i \leq x \leq x_{i+1} \end{cases} \quad (2.8)$$

θ is a constant introduced so as to make (θ/D^-) and (θ/D^+) close to 1, for numerical reasons.

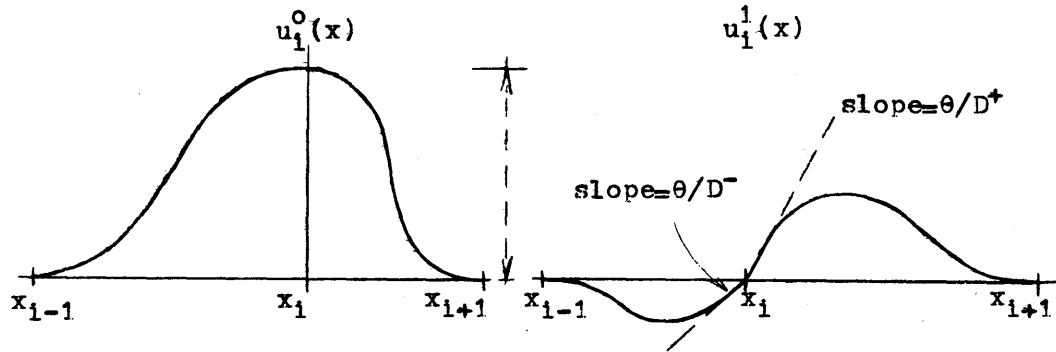


Fig.2.4 - Cubic basis functions appropriate for flux and current continuity across interfaces for a neutron diffusion problem.

In the case of piecewise linear functions we will be able to preserve function continuity across interfaces. The element functions are expressed by Eq. (2.9)

$$u_{i+1}^{0-}(x) = \begin{cases} \frac{x-x_i}{x_{i+1}-x_i} & , x_i \leq x \leq x_{i+1} \\ 0 & , \text{otherwise} \end{cases} \quad (2.9.a)$$

$$u_i^{0+}(x) = \begin{cases} \frac{x_{i+1}-x}{x_{i+1}-x_i} & , x_i \leq x \leq x_{i+1} \\ 0 & , \text{otherwise} \end{cases} \quad (2.9.b)$$

When the proper coupling conditions (function continuity) are imposed we will have defined the basis functions for the expansion, as shown in Fig. 2.5.

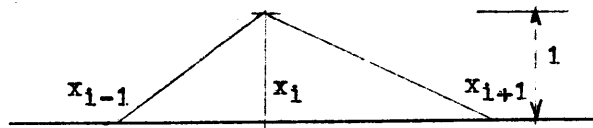


Fig. 2.5 - Linear basis function appropriate for flux continuity across interfaces.

It is shown in ref. 7 that, if the solution is approximated through the "weak" form, and this "weak" form is positive definite, then the error bounds of a neutron diffusion source problem are

$$\|f(x) - s(x)\| \leq K \overline{\Delta x}^{2m}$$

where $\overline{\Delta x}$ is an upper bound of the mesh sizes, and K is a constant independent of $\overline{\Delta x}$.

So, in the cases considered we will have the following error bounds.

$$\begin{aligned} \|f(x) - s(x)\| &\leq K_l \overline{\Delta x}^2 && \text{(linear)} \\ &\leq K_c \overline{\Delta x}^4 && \text{(cubic)} \end{aligned}$$

2.3) The basis functions in two dimensions

The element functions in two dimensions are generated by taking products of the one-dimensional ones in each direction. For instance, the two-dimensional bicubic interpolate in one element (see Fig. 2.6, element IV) is given by:

$$\begin{aligned} s_i(x, y) = \sum_{px=0}^1 \sum_{py=0}^1 &\left[f_{i,j}^{(px+py)} u_i^{px+}(x) u_j^{py+}(y) + f_{i+1,j}^{(px+py)} u_{i+1}^{px-}(x) \right. \\ &u_j^{py+}(y) + f_{i,j+1}^{(px+py)} u_i^{px+}(x) u_{j+1}^{py-}(y) + \\ &\left. f_{i+1,j+1}^{(px+py)} u_{i+1}^{px-}(x) u_{j+1}^{py-}(y) \right] \end{aligned}$$

where $f_{i,j}$ means the value of the function at point (i, j) , and

where the superscript $(px+py)$ on f means the px^{th} derivative in the X direction and the py^{th} derivative in the Y direction.

There are, consequently, 16 values to be known at the 4 corner points: function values, the first derivative in each direction, and the 4 mixed derivatives $\partial^2/\partial x \partial y$.

It is shown in refs. 6 and 7 that a two-dimensional polynomial interpolate $s_1(x,y)$ is uniquely determined, given the appropriate corner values, and for sufficiently smooth functions satisfies the following pointwise error bound in any norm.

$$\left\| \frac{\partial^{(px+py)}}{\partial^{px} \partial^{py}} (f(x,y) - s(x,y)) \right\| \leq K_x \Delta x^{(2mx-px)} + K_y \Delta y^{(2my-py)}$$

$$\text{for } 0 \leq px \leq mx-1$$

$$\text{and } 0 \leq py \leq my-1$$

Where, K_x and K_y are positive constants independent of Δx and Δy respectively, and where $(2mx-1)$ and $(2my-1)$ specify the degree of the one-dimensional polynomials in each direction.

Consequently, in the case of bilinear and bicubic interpolation we have the following interpolation error bounds:

$$\left\| f(x,y) - s(x,y) \right\| \leq K_1 (\Delta x^2 + \Delta y^2) \quad (\text{linear})$$

$$\leq K_c (\Delta x^4 + \Delta y^4) \quad (\text{cubic})$$

where K is an upper bound of K_x and K_y .

As in the one-dimensional case, basic functions for neutron diffusion problems can be defined by imposing appropriate coupling conditions to the element functions. The unknown solution of a two-dimensional problem can then be approximated by considering a convenient partition of the problem domain.

We will show the procedure for construction of the basis functions for the case of bicubic element functions.

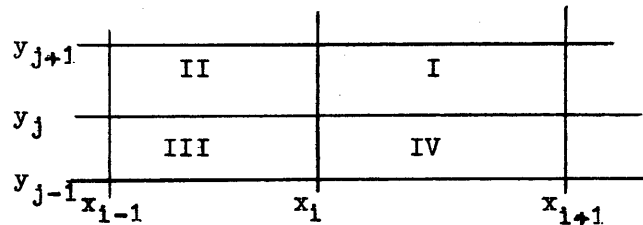


Fig. 2.6 - Two-dimensional partition

Case 1

$$D_I D_{III} = D_{II} D_{IV}$$

This condition is satisfied at any interior or interface points, except singular points. The basis functions which allow current or simply derivative continuity across the i and j lines are Eq. (2.10)

$$u_{ij}^{00}(x,y) = \left[\begin{array}{ll} u_i^{0+}(x) u_j^{0+}(y) & , \quad \text{I} \\ u_i^{0-}(x) u_j^{0+}(y) & , \quad \text{II} \\ u_i^{0+}(x) u_j^{0-}(y) & , \quad \text{III} \\ u_i^{0-}(x) u_j^{0-}(y) & , \quad \text{IV} \end{array} \right. \quad (2.10a)$$

$$u_{ij}^{10}(x,y) = \left[\begin{array}{ll} \frac{\theta}{D_I} u_i^{1+}(x) u_j^{0+}(y) & , \quad \text{I} \\ \frac{\theta}{D_{II}} u_i^{1-}(x) u_j^{0+}(y) & , \quad \text{II} \\ \frac{\theta}{D_{III}} u_i^{1+}(x) u_j^{0-}(y) & , \quad \text{III} \\ \frac{\theta}{D_{IV}} u_i^{1-}(x) u_j^{0-}(y) & , \quad \text{IV} \end{array} \right. \quad (2.10b)$$

$$u_{ij}^{01}(x,y) = \left[\begin{array}{ll} \frac{\theta}{D_I} u_i^{0+}(x) u_j^{1+}(y) & , \quad \text{I} \\ \frac{\theta}{D_{II}} u_i^{0-}(x) u_j^{1+}(y) & , \quad \text{II} \\ \frac{\theta}{D_{III}} u_i^{0+}(x) u_j^{1-}(y) & , \quad \text{III} \\ \frac{\theta}{D_{IV}} u_i^{0-}(x) u_j^{1-}(y) & , \quad \text{IV} \end{array} \right. \quad (2.10c)$$

$$u_{ij}^{11}(x,y) = \left[\begin{array}{ll} \frac{\theta}{D_I} u_1^{1+}(x) u_j^{1+}(y) & , \text{ I} \\ \frac{\theta}{D_{II}} u_1^{1-}(x) u_j^{1+}(y) & , \text{ II} \\ \frac{\theta}{D_{III}} u_1^{1+}(x) u_j^{1-}(y) & , \text{ III} \\ \frac{\theta}{D_{IV}} u_1^{1-}(x) u_j^{1-}(y) & , \text{ IV} \end{array} \right. \quad (2.10d)$$

In these equations, θ is a normalization constant chosen so that $\theta/D = 1$.

Case 2

$$D_I D_{III} \neq D_{II} D_{IV} \quad (\text{singular points})$$

It is easy to see that at singular points the current continuity condition of the diffusion problem can be met by specifying

$$\frac{\partial}{\partial x} \phi(x,y) = \frac{\partial}{\partial y} \phi(x,y) = 0$$

for (x,y) - singular point

However, we know that the analytic solution at a singular point does not satisfy the above conditions. Further some results to be shown later indicate that imposition of the above condition gives poor approximate solutions.

As mentioned previously, diffusion theory itself is not valid at corners with discontinuities. We do not expect analytic

solutions near singularities to truly represent the physical neutron distribution. Furthermore, on a practical point of view, in many problems one is more interested in overall measures of core performance (k_{eff} , region-wise reaction rates). In these cases one may argue that, whatever the analytic singular component of the solution, it is a localized effect and its contribution to overall features should be very small.

On the other hand, in real life two-dimensional calculations one will have some kind of scattered pattern of zones with different enrichments inside a reactor, so that most of the actual points will be singular ones. We expect the singularity to be a function of the magnitude of the material discontinuities. It will have a greater influence at water-fuel interfaces than at interfaces between fuels of different enrichments.

One of the purposes of the present thesis was to investigate how one might proceed in dealing with singular points in problems where one is mainly concerned with regionwise, averaged behaviors (such as a depletion analysis). One might, for instance, arbitrarily specify first derivative continuity by making all $\theta/D = 1$ at Eq. (2.12b) and (2.12c). This approach seems intuitively convenient for "fuel only" singular points, where the diffusion coefficients do not change drastically.

On the other hand one may also uncouple completely the first derivatives by "opening" Eq. (2.10b) and (2.10c) into four equations.

$$u_{ij}^{\bar{1}0}(x,y) = \begin{cases} u_i^{1-}(x) u_j^{0+}(y) & , & \text{II} \\ u_i^{1-}(x) u_j^{0-}(y) & , & \text{III} & (2.10b-) \\ 0 & , & \text{I \& IV} \end{cases}$$

$$u_{ij}^{+10}(x,y) = \begin{cases} u_i^{1+}(x) u_j^{0+}(y) & , & \text{I} \\ u_i^{1+}(x) u_j^{0-}(y) & , & \text{IV} & (2.10b+) \\ 0 & , & \text{II \& III} \end{cases}$$

$$u_{ij}^{0\bar{1}}(x,y) = \begin{cases} u_i^{0-}(x) u_j^{1-}(y) & , & \text{III} \\ u_i^{0+}(x) u_j^{1-}(y) & , & \text{IV} & (2.10c-) \\ 0 & , & \text{I \& II} \end{cases}$$

$$u_{ij}^{+0\bar{1}}(x,y) = \begin{cases} u_i^{0+}(x) u_j^{1+}(y) & , & \text{I} \\ u_i^{0-}(x) u_j^{1+}(y) & , & \text{II} & (2.10c+) \\ 0 & , & \text{III \& IV} \end{cases}$$

$$u_{ij}^{00}(x,y) - \text{Eq. (2.10a)}$$

$$u_{ij}^{11}(x,y) - \text{Eq. (2.10d)}$$

Still another possibility would be to impose the zero derivatives conditions by dropping Eq. (2.10b) and (2.10c) completely from the set of expansion functions. This approach, however, leads to intolerable distortions both in flux shape near the singular points and in the overall characteristics.

In Ch. III we will show results arrived at for different problems and ways of dealing with singular points.

The generation of basic functions at the boundaries of the domain can be easily shown to be special cases of the expressions given by Eq. (2.10), arrived at by coupling element functions whose supports are non-zero.

As in the one-dimensional case, refs. 6 and 7 show that, if the solution to the diffusion problem is approximated through the weak form, and the weak form is positive definite, then the error bounds of a neutron diffusion source problem, for $f(x,y)$ sufficiently differentiable, are

$$\| f(x,y) - s(x,y) \|_{\infty} \leq K_x \overline{\Delta x}^{2m} + K_y \overline{\Delta y}^{2m}$$

Where $\overline{\Delta x}$ and $\overline{\Delta y}$ are upper bounds of the mesh sizes in each direction, and K_x and K_y are constants independent of $\overline{\Delta x}$ and $\overline{\Delta y}$ respectively.

In the bilinear and bicubic cases consequently

$$\begin{aligned} \| f(x,y) - s(x,y) \|_{\infty} &\leq K_1 (\overline{\Delta x}^2 + \overline{\Delta y}^2) \text{ (linear)} \\ &\leq K_c (\overline{\Delta x}^4 + \overline{\Delta y}^4) \text{ (cubic)} \end{aligned}$$

Where the K are convenient upper bounds of K_x and K_y in each case.

2.4) Varying cross sections inside the expansion element

Up to this point we have tacitly assumed that the diffusion coefficient was a piecewise constant function. The weak form of the problem is properly posed for more general behavior of the diffusion coefficient. In particular, a unique solution to the problem will exist if $D(\underline{r})$ is allowed to be a piecewise continuous, rather than piecewise constant, function.

From a theoretical view piecewise continuous diffusion coefficients present no difficulties. From the view of the approximation methods the inclusion merely changes the evaluation of elements in the coefficient matrix.

The number of basis functions (and consequently the number of unknowns) is approximately equal to some integer times the number of mesh points. We have assumed that mesh lines are always located at interfaces. If the system has many discontinuities then we will have many unknowns. It is evident that if the spacing between discontinuities is very small, then finite difference methods may be adequate for the problem, and one need not consider higher-order methods. Conversely, it is of interest to attempt to use coarse meshes, and higher order methods, in an attempt to find accurate solutions with a small computing burden. Clearly then, for highly heterogeneous systems, we must consider the problem of mesh intervals which extend through discontinuities. It is just this problem that we consider in some detail in this thesis.

Our approach will be the following: We will use element and/or basis functions which are themselves smooth within their interval of definition. Discontinuities in properties, within the interval, will be accounted for in all inner products. The approximate solution will be smoother than the actual solution; however the effects of the discontinuities will be averaged into the approximate solution through the elements of the coefficient matrix.

2.5) Matrix formulation of the two-group diffusion problem and numerical methods

In this section we present the matrix equation which results from the application of the finite element method to the two-group diffusion problem. We further present the numerical methods used in the actual solution of the system with the computer code CHD. We will exemplify the procedure with a simple, two-dimensional, two-group problem, with the realization that an extension to more groups is straightforward.

We consider a partition of the problem domain in Cartesian coordinates on a "bone fide" basis. In a PWR, for instance, a partition along fuel element interfaces will provide surprisingly accurate results. Let N be the number of rows and M be the number of columns resulting from the partition.

We next preassign some order to the unknowns and define two column vectors \underline{a}_1 and \underline{a}_2 as the ordered set of the unknown solutions in each group. Although it is not crucial, the form of the matrices is easier to visualize and it will be assumed in this

thesis that the ordering is done by assigning to each point all its associated unknowns (flux, first derivatives, mixed derivative for bicubic expansion, for instance) in some order, before moving to the next point.

Consider now Eq. (1.2) specialized for two dimensions and two groups

$$A_1 \phi_1(x,y) = \frac{1}{\lambda} [v \Sigma_{f1} \phi_1(x,y) + v \Sigma_{f2} \phi_2(x,y)]$$

$$A_2 \phi_2(x,y) = \Sigma_{s1} \phi_1(x,y)$$

where the spatial variation of the operators is implied.

We next approximate the fluxes by means of the piecewise Hermite polynomials (u) defined in the previous sections:

$$\phi \approx \hat{\phi} = \sum_{n=1}^N \sum_{m=1}^M \sum_{i=1}^{I(n,m)} a_{n,m,i}^1 u_{n,m,i}^1(x,y)$$

$$\phi \approx \hat{\phi} = \sum_{n=1}^N \sum_{m=1}^M \sum_{i=1}^{I(n,m)} a_{n,m,i}^2 u_{n,m,i}^2(x,y)$$

where $I(n,m)$ is the number of basis functions at point (n,m) , and where $N, M, I(n,m)$ are taken as being the same for both groups for simplicity of notation.

We seek the approximate solutions in the coupled "weak" form of the two-group equations. The Galerkin method requires that we apply Eq. (2.5), the overall result being that the following set of coupled linear equations be solved.

$$\int_V \left[D_1 \nabla \hat{\phi}_1 \cdot \nabla u_{n,m,i}^1 + [\Sigma_{r1} \hat{\phi}_1 - \frac{1}{\hat{\lambda}} (v \Sigma_{f1} \hat{\phi}_1 + v \Sigma_{f2} \hat{\phi}_2)] u_{n,m,i}^1 \right] dV = 0$$

$$\int_V \left[D_2 \nabla \hat{\phi}_2 \cdot \nabla u_{n,m,i}^2 + [\Sigma_{r2} \hat{\phi}_2 - \Sigma_{s1} \hat{\phi}_1] u_{n,m,i}^2 \right] dV$$

$$\begin{aligned} \text{for } n &= 1, N \\ m &= 1, M \\ i &= 1, I(n, m) \end{aligned} \quad (2.11)$$

Where $\hat{\lambda}$ is introduced as an approximation to λ .

When the necessary products of the Hermite functions are performed, we end up with two coupled matrix equations of the form

$$\underline{A}_1 \underline{a}_1 = \frac{1}{\hat{\lambda}} [\underline{F}_1 \underline{a}_1 + \underline{F}_2 \underline{a}_2]$$

$$\underline{A}_2 \underline{a}_2 = \underline{S}_1 \underline{a}_1 \quad (2.12)$$

Where the \underline{A} are matrices which incorporate the diffusion and removal terms, the \underline{F} incorporate the fission terms and \underline{S}_1 incorporates the group 1 downscattering terms.

It is useful to focus some attention on the forms of the matrices of Eq. (2.12). They will all show an overall stripe structure, the number of stripes depending on the dimensions of the problem.

$$\text{Number of stripes} = 3^k$$

$$k = \text{No. of spatial dimensions (1, 2 or 3)}$$

Additionally, the stripes themselves are formed by block matrices whose size will depend on the number of coupling expansion functions associated with the relevant points.

In Fig. (2.7) we show the general form of these block matrices for a two-dimensional problem. Each position $(a,b...i)$ represents a block matrix, the size depending on the number of basis functions existing at each of the coupling points. The coupling between a general point (n,m) and its immediate neighbors (and with itself) is shown.

In Fig. (2.8) the structure of one of the A matrices is shown for a simple 3×3 two-dimensional partition. The problem is too small so that the stripes are not evident and we show just the limits of the diagonal.

It can be shown (ref. 6 and 7) that the matrices \underline{A}_1 and \underline{A}_2 are always symmetric and positive definite. \underline{F}_2 and \underline{S}_1 are not symmetric in the general case and, in fact, may even not be square if the total number of expansion functions in the fast group is not the same as in the thermal group. Matrix \underline{F}_1 is also always symmetric.

We choose to solve Eq.(2.12) by uncoupling the system through the power iteration method (ref. 11). This method has been used successfully in conjunction with finite difference methods and can be shown to yield, in these cases, solutions which are everywhere positive (the fundamental modes) for any positive initial guess, and the corresponding eigenvalue which can thus be identified with the k_{eff} of the system. On practical grounds we have reasons to expect that the same will be true in the present case, for any initial guess corresponding to everywhere positive fluxes.

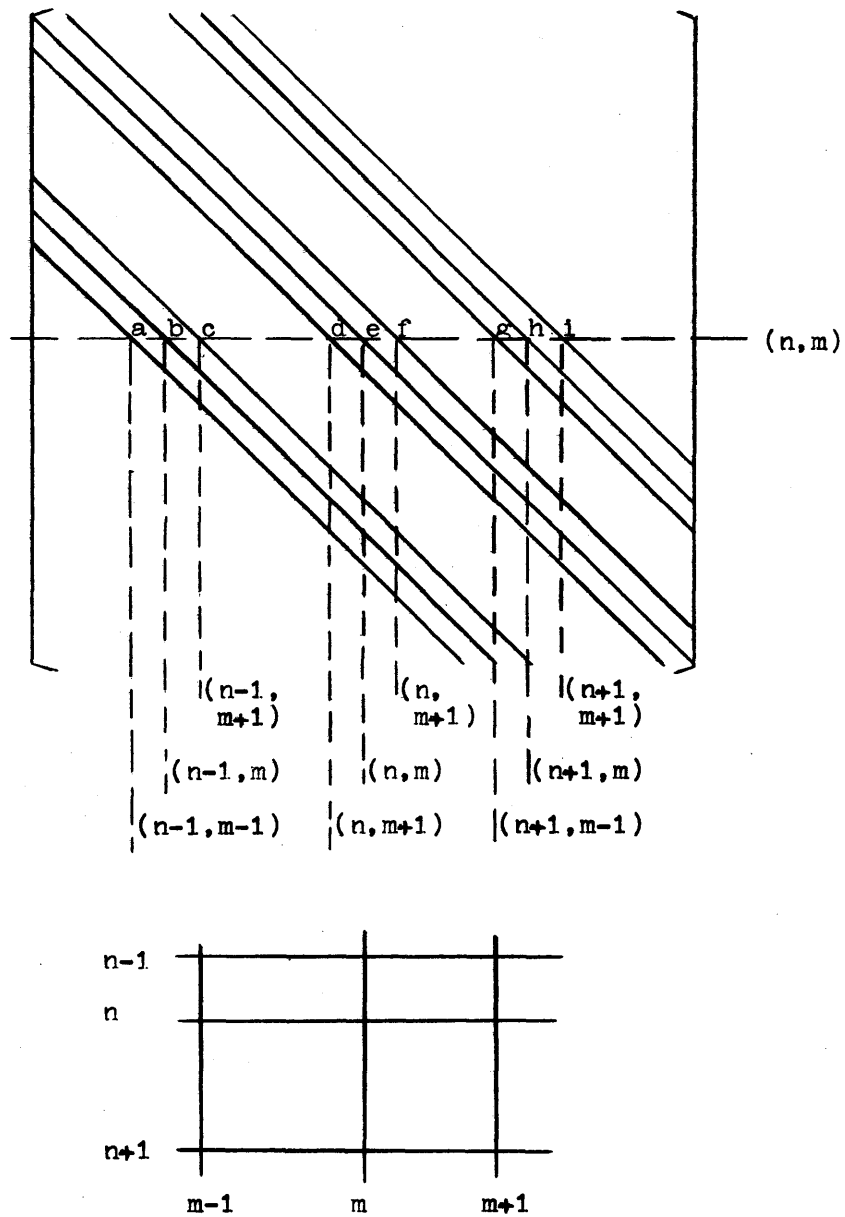


Fig. 2.7 - Two-dimensional partition and matrix form for unknowns ordered as: 1) all unknowns at each point; 2) all columns; 3) all rows.

The power iteration method requires the following iteration procedure.

$$\underline{A}_1 \underline{b}_1^{(t)} = \underline{F}_1 \underline{a}_1^{(t-1)} + \underline{F}_2 \underline{a}_2^{(t-1)}$$

$$\underline{A}_2 \underline{b}_2^{(t)} = \underline{S}_1 \underline{b}_1^{(t)}$$

$$\hat{\lambda}^{(t)} = \frac{(\underline{b}_1^{(t)}, \underline{b}_1^{(t)}) + (\underline{b}_2^{(t)}, \underline{b}_2^{(t)})}{(\underline{b}_1^{(t)}, \underline{a}_1^{(t-1)}) + (\underline{b}_2^{(t)}, \underline{a}_2^{(t-1)})}$$

$$\underline{a}_1^{(t)} = \frac{\underline{b}_1^{(t)}}{\hat{\lambda}^{(t)}}$$

$$\underline{a}_2^{(t)} = \frac{\underline{b}_2^{(t)}}{\hat{\lambda}^{(t)}}$$

(2.13)

Where the parenthesis indicate inner products.

We next will show the numerical scheme used to solve each of Eq. (2.13) that is, the problems

$$\underline{b}_1^{(t)} = \underline{A}_1^{-1} [\underline{F}_1 \underline{a}_1^{(t-1)} + \underline{F}_2 \underline{a}_2^{(t-1)}]$$

$$\underline{b}_2^{(t)} = \underline{A}_2^{-1} \underline{S}_1 \underline{b}_1^{(t)}$$

When the matrices \underline{A}_1 and \underline{A}_2 are positive definite and and not too large the Cholesky factorization scheme (ref. 12) can be conveniently utilized^(*). It is shown that in this case the A's can be factorized as

$$A = \underline{E} \underline{E}^T$$

where \underline{E} is a lower triangular matrix.

The elements of \underline{E} can be determined by using the algorithm

$$e_{jj} = [a_{jj} - \sum_{k=1}^{j-1} e_{jk}^2]^{1/2}$$

$$e_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} e_{ik} e_{jk}}{e_{jj}}$$

The problem can then be solved in two sweeps, one forward and one backward to insert \underline{E} and \underline{E}^T respectively.

(*) Matrices of up to 350 x 350 elements have been inverted without problems in double precision.

CHAPTER III
SAMPLE PROBLEMS AND RESULTS

The final aim of this chapter is to present some results on the approximate solution of realistic, two-dimensional reactor problems as they were evaluated with CHD. The most important example chosen was the first-loading, beginning-of-life core of the Zion 1 reactor (ref.13) the evaluation being done with 3 different kinds of coarse meshes, on a quarter-core symmetry basis. The results are compared to conventional finite difference results, as obtained with the use of the EXTERMINATOR 2, PDQ-5 and CITATION codes. For the finite element examples the finest mesh used corresponds to a full fuel element (about 22 cm). Subsequently, some meshes were defined over 4 fuel elements and, finally, a very coarse mesh was defined over the innermost 16 fuel elements.

Several questions arise when coarse meshes are used. For instance, when two regions with varying cross sections have a common interface, it is not at all clear whether the best continuity condition should actually preserve current across the interface. The same reasoning is actually valid for any mesh point surrounded by regions with varying parameters.

Even in the simplest case, when cross sections are constant inside the expansion elements, it is not clear which conditions should be imposed across singular points. Zero first derivative conditions, for instance, are known to preserve the appropriate continuity conditions of the diffusion problem. However, it is

also known that the diffusion approximation itself does not hold in the vicinity of these points, so that it may be appropriate to relax such conditions.

All these questions were investigated through a set of appropriately defined smaller problems. In some cases the results are compared with finite difference results, through the use of the EXTERMINATOR 2 code, and in other cases they are compared to very fine mesh results of CHD, which are known to be very accurate.

The order in which the results are presented in the following sections corresponds to an increasing complexity of the problems.

All cross sections are listed in Appendix A.

3.1) Comparisons between CHD and EXTERMINATOR 2 for a 4-composition problem.

In its earliest versions, CHD was tested against EXTERMINATOR 2 for the small problem shown in Fig. 3.1. The basic 18 cm measure corresponds approximately to the width of a PWR fuel element.

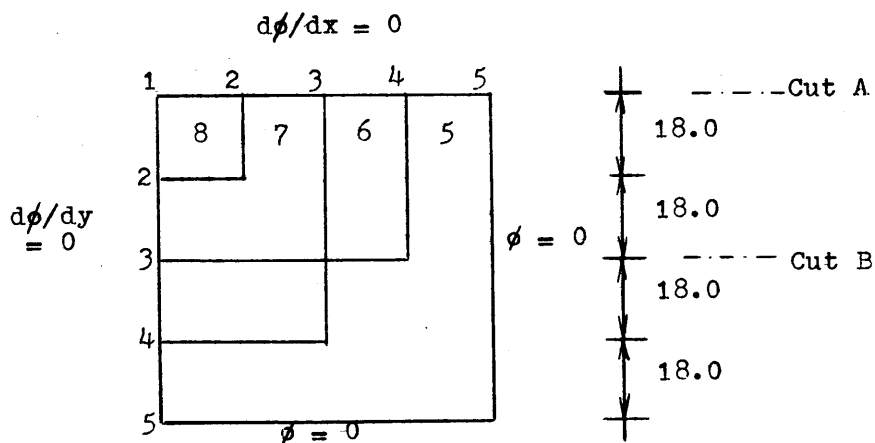


Fig. 3.1 - The 4-composition problem geometry

EXTERMINATOR 2 was developed at Oak Ridge, and the version used at MIT corresponds essentially to ORNL 4078 (ref. 14). It is a standard finite difference code, using periodic extrapolation as acceleration technique for the outer iterations, and row or column successive overrelaxation in the inner iterations.

Several EXTERMINATOR 2 and CHD runs were made, the purpose being mainly to investigate what kind of derivative continuity condition (if any) should be used at singular points such as 22, 33, 43 and 34. It was to be expected beforehand that at singular points surrounded by fuel elements one could get sufficiently accurate results by requiring simply derivative continuity, because the diffusion coefficients are not very different for homogenized fuel elements of different enrichments. For singular points where there are fuel-water interfaces one might expect that uncoupling the first derivatives should improve the results. Continuities imposed directly on the expansion functions result in less unknowns, whereas each uncoupling increases the number of unknowns by 1. Thus the problem consists basically in determining if the imposition of continuities will not distort the results in a way intolerable for the purpose of the calculation.

Various combinations of element functions were used throughout the work. They are identified by code numbers, and are described in detail in App. C.

Macroscopic two-group cross sections were taken from ref. 10, Table 5.1, and are close to low, medium and highly enriched, homogenized PWR fuel element properties.

In Table 3.1 we compare overall features for all the runs.

Table 3.1 Overall results for EXTERMINATOR 2 and CHD for the 4-composition problem

Case	Code (mesh.cm)	Solution condition at singular points	# of unknowns per group	k_{eff}	Average fluxes at reentrant water region (fast/th.)
	EXTERM. (9.0)	-	81	1.09472	
	EXTERM. (4.5)	..	289	1.08260	
	EXTERM. (3.0)	-	625	1.07972	
Ref.	EXTERM. (3.0 & 1.8)	-	1156	1.07840	0.25973 0.44608
1	CHD (18.0)	Zero first derivatives	56	1.07598	0.30615 0.45573
2	CHD (18.0)	First deri- vatives con- tinuity. (41 sets)	64	1.07825	0.25740 0.44611
3	CHD (18.0)	First deri- vatives un- coupled. (61 sets)	72	1.07812	0.25564 0.44631

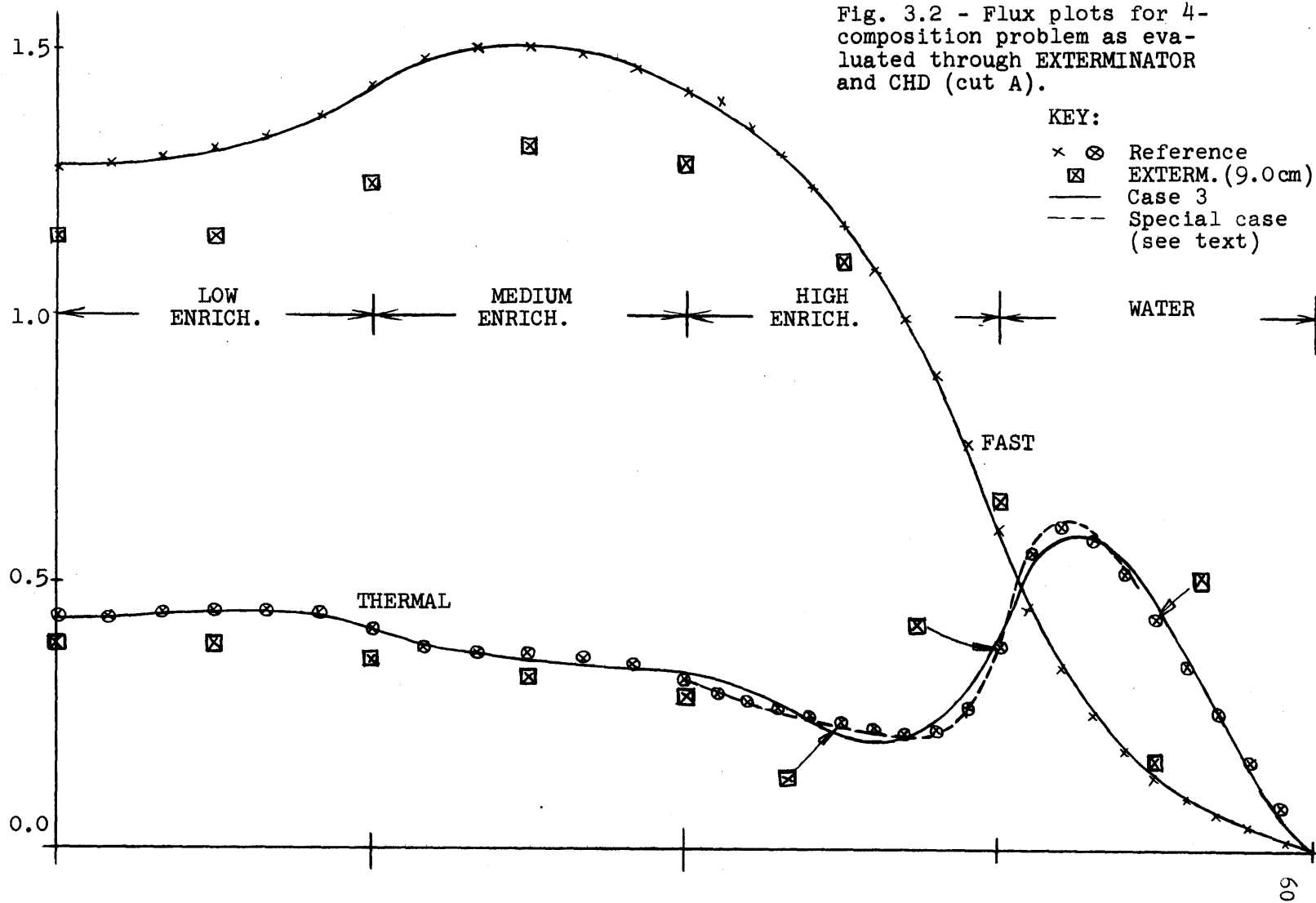
It can be seen that, except for the case of zero derivatives at all singular points, we can reproduce k_{eff} up to four decimal places with CHD, using a whole fuel element as a mesh and thus requiring much fewer unknowns, in general by a factor over 10. On the other hand, the significant deviation of the EXTERMINATOR result from the reference k_{eff} , as the finite difference mesh is increased, is also very clear. In particular, the 9.0 cm EXTERMINATOR mesh requires 81 unknowns per group, about the same as all the CHD runs: whereas the finite element method yields k_{eff} within less than 0.03% in cases 2 and 3, the finite difference result is off by 1.63%.

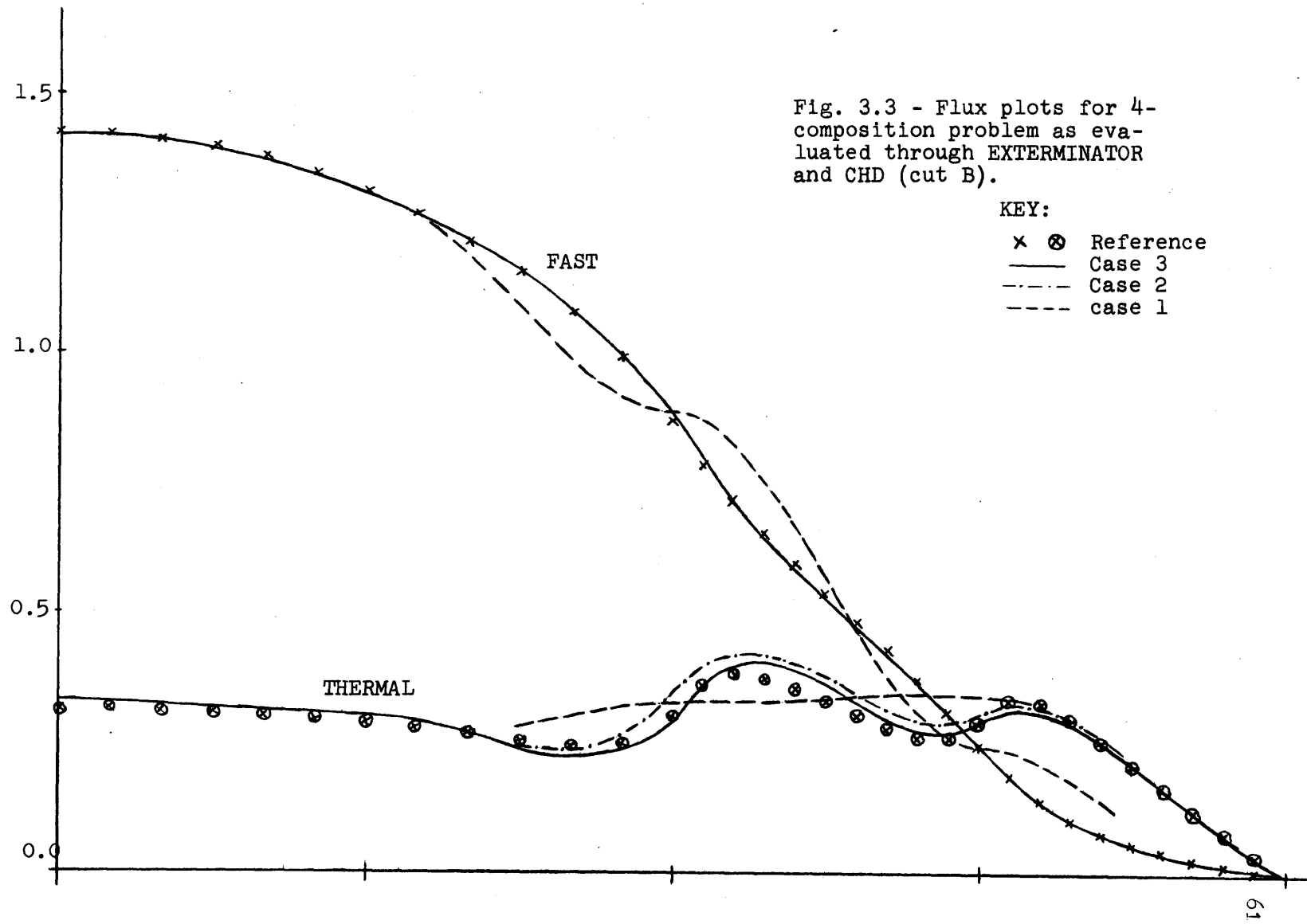
We have also included in the last column a measure of the average fluxes at the reentrant water regions, where one may expect the largest errors to occur. In the finite element cases 2 and 3 fast fluxes are represented within less than 1.6% whereas thermal fluxes are represented within less than 0.05%.

It should also be realized that the distortion due to the singular points is a very localized effect and its influence on the overall core performance (exemplified here by k_{eff}) is rather small.

In Figs. 3.2 and 3.3 we have plotted fast and thermal fluxes for cuts through positions A and B, and in Fig. 3.4 we present element-averaged results.

It is evident from Fig. 3.3 that case 1, where we have imposed zero derivative across the singular points, is not a realistic assumption and does not yield accurate pointwise results. For this reason, none of the following examples was ever evaluated with this option.





11.0455 1.36019 0.43627	12.3315 1.45922 0.35520	10.0330 1.08427 0.25101
	10.8738 1.30455 0.31100	8.4419 0.83480 0.21987
		1.58602 0.25973 0.44608

Reference (EXTERMINATOR)

11.022 1.3556 0.43558	12.265 1.4523 0.35318	9.9761 1.0755 0.24986
	10.959 1.2653 0.31958	8.4350 0.81152 0.22220
		1.6245 0.30615 0.45573

Case 1 (CHD)

11.041 1.3568 0.43651	12.305 1.4606 0.35385	10.033 1.0843 0.25098
	10.863 1.3069 0.31023	8.4753 0.83484 0.22103
		1.5860 0.25740 0.44611

Case 2 (CHD)

11.055 1.3585 0.43707	12.321 1.4621 0.35437	10.048 1.0851 0.25146
	10.849 1.3071 0.30959	8.4510 0.83493 0.22017
		1.5866 0.25564 0.44631

Case 3 (CHD)

KEY

Absorp. rate
Avg. fast flux
Avg. ther. flux

Fig. 3.4 - Element averaged results for 4-composition problem as evaluated with EXTERMINATOR and CHD

In case 3 we uncouple the first derivatives at all singular points. It is clear that the pointwise fluxes are better matched in this case. Further the element-averaged results are well-matched in case 2, where first derivative continuity is imposed across the singular points, as well as in case 3.

In Fig. 3.2 we have also included a thermal flux plot labeled (S) for Special Case, at the fuel water interface. For this case two extra mesh points were included at each side of the fuel water interface at $x = 0.0, 4.5$ cm to each side of the interface. The pointwise error is negligible in this case. Clearly thus we can improve local accuracy by refining the mesh slightly in the region of large gradients.

3.2) The one-dimensional, ten-region problem

When one tries to represent a core such as Zion 1 on a coarse mesh, the core baffle arises as an anomaly perturbing the regular structure of the fuel elements, which would otherwise provide an excellent partition for the problem. The explicit representation introduces undesired fine meshes at several points of the region.

On the other hand, the baffle has a relatively strong influence on the system (mainly in the thermal flux distribution of adjacent elements) so that it cannot be neglected. In order to account for the baffle's presence in the finite element method (an inherently coarse mesh method) one can either homogenize it with some portion of the reflector, or define a region with varying parameters on the outside of the core, which would include the baffle and water.

In order to investigate how accurate a solution one could get by homogenizing the baffle in a real reactor, a series of one-dimensional (*) runs were made with the dimensions, compositions and number of regions of the Zion 1 reactor. The geometry shown in Fig. 3.4a was used, with cross sections corresponding to values arrived at for these fuel elements (ref. 13).

All the results presented were obtained from CHD.

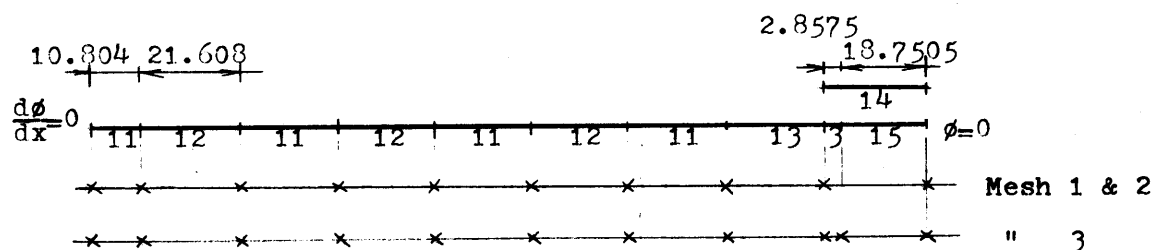


Fig. 3.4a - One-dimensional, ten-region problem.

In Tables 3.2 and 3.3 we present some overall results, and in Fig. 3.5 we present flux plots. It can be seen, in general, that the baffle causes relatively important changes in the system. Although k_{eff} can be fairly well corrected by smearing the SS into the adjacent reflector region, pointwise flux deviations of 100% from the correct answer occur at fuel-baffle interface.

(*) Actually, CHD does not have a one-dimensional option incorporated. However, 1D can be simulated by using zero derivative conditions at two opposite boundaries of a convenient two-dimensional problem.

Table 3.2 - Eigenvalues for one-dimensional, ten-region problem.

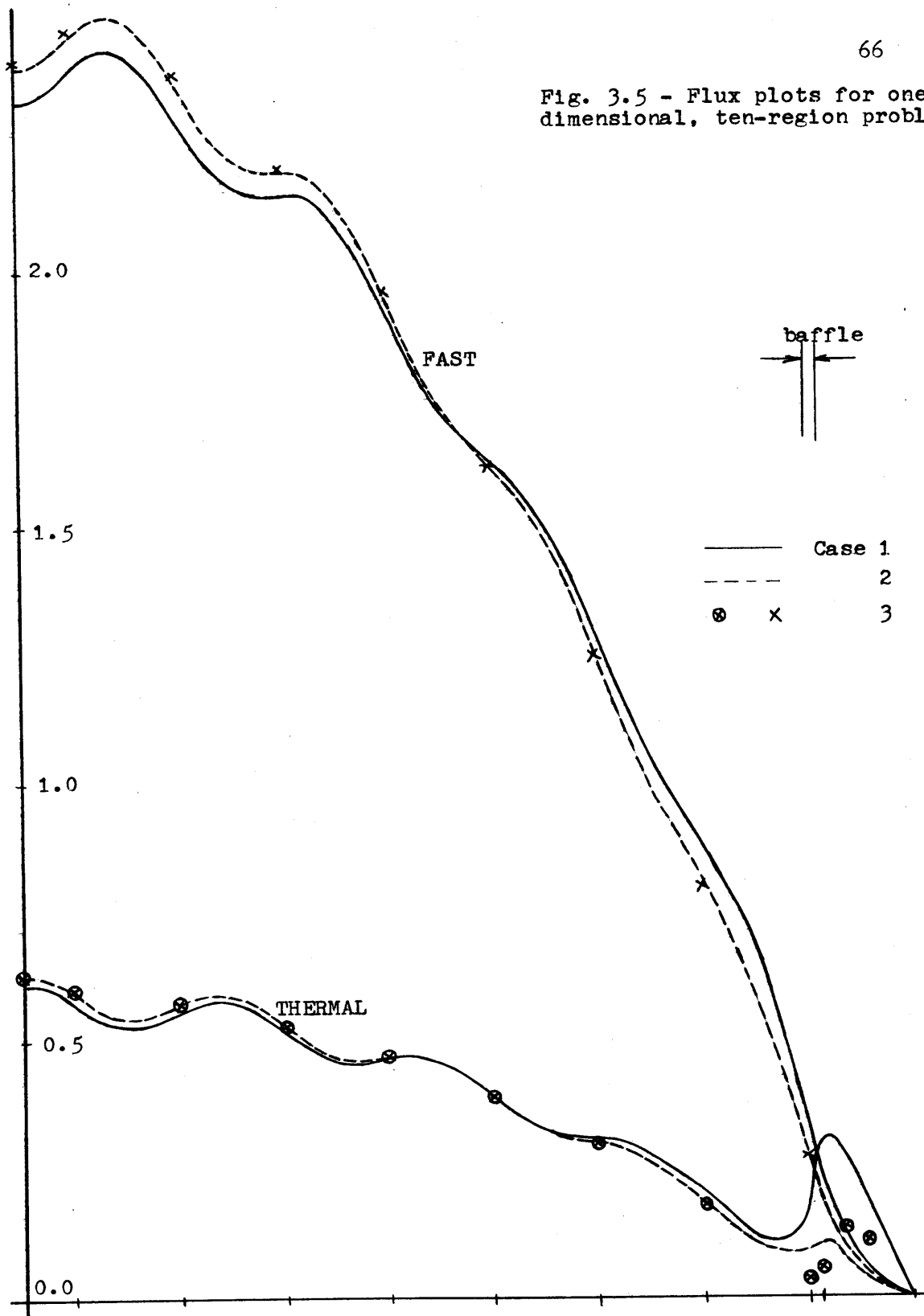
Case	1	2	3
Description	No baffle	Baffle and water homogenized.	Baffle explicitly considered.
k_{eff}	1.27912	1.27873	1.27866

Table 3.3 - Element averaged results for one-dimensional, ten-region problem.

Case	Composition								
	11	12	11	12	11	12	11	13	¹⁵ 14 3 & 15
1	7.02	14.58	13.13	12.68	10.48	9.07	6.38	4.28	0.427
	0.602	0.543	0.562	0.472	0.450	0.338	0.273	0.148	
2	7.23	15.00	13.45	12.90	10.54	8.94	6.06	3.58	0.431
	0.620	0.558	0.576	0.480	0.451	0.333	0.259	0.121	
3	7.28	15.09	13.53	12.96	10.56	8.93	6.01	3.37	0.378
	0.624	0.562	0.579	0.483	0.452	0.333	0.257	0.112	

KEY Absorp. rate
Thermal flux

Fig. 3.5 - Flux plots for one-dimensional, ten-region problem.



From these runs we draw the conclusion that for a real reactor one can probably arrive at sufficiently accurate results, as far as average properties are concerned, by smearing the baffle in the nearby moderator region. The coarse mesh structure can thus be maintained throughout the problem domain.

Pointwise flux results however show significant deviations which one would like to reduce. Therefore, we will try to account for the baffle's presence by incorporating it into a region with varying parameters together with some portion of the nearby reflector.

In the following sections all the implications that arise when defining regions with varying parameters will be evaluated through small, two-dimensional problems. First, ways will be shown to deal with cases such as the baffle, when one is interested just in the influence of an outside region on a core and not in the detailed flux behavior inside the perturbing region. Second, solutions will be shown for cases in which varying cross sections are specified inside a fuel region .

We begin by showing in Sec. (3.3) a simple two-dimensional problem, in which modifications will be introduced in Sec. (3.4) and (3.5) to allow investigation of the problems we have mentioned.

3.3) The basic two-dimensional two-composition problem

One of the sample two-dimensional problems used in ref. 7 was the two-composition core shown in Fig. 3.6.

In ref. 7 a fine finite difference result (2 cm mesh) is compared with results from Hermite polynomials expansions for the cases of linear and cubic basis functions and, in the latter case, for 3 ways of taking into account the singularity. Some of the k_{eff} are shown in Table 3.4, and in Figs. 3.7 we show thermal flux plots for $x = 0$ and $x = 20$ cm.

Set A of the expansion functions is defined so as to preserve currents across the singular point by making

$$\frac{\partial \phi}{\partial x} = \frac{\partial \phi}{\partial y} = 0 \quad \text{for } x = y = 20 \text{ cm}$$

As we have already seen, this option is not interesting in actual practice, because the zero derivatives introduce great distortions in the flux shapes. It is no longer an option in CHD.

Set B specifies derivative continuity across the singular point and corresponds to option 41 in CHD.

Set C uncouples the first derivatives and corresponds to option 61 in CHD.

As was to be expected, the best pointwise flux distribution was achieved with set C, although for a 20 cm mesh none of the solutions

Fig. 3.6 - The basic two-dimensional, two-composition problem.

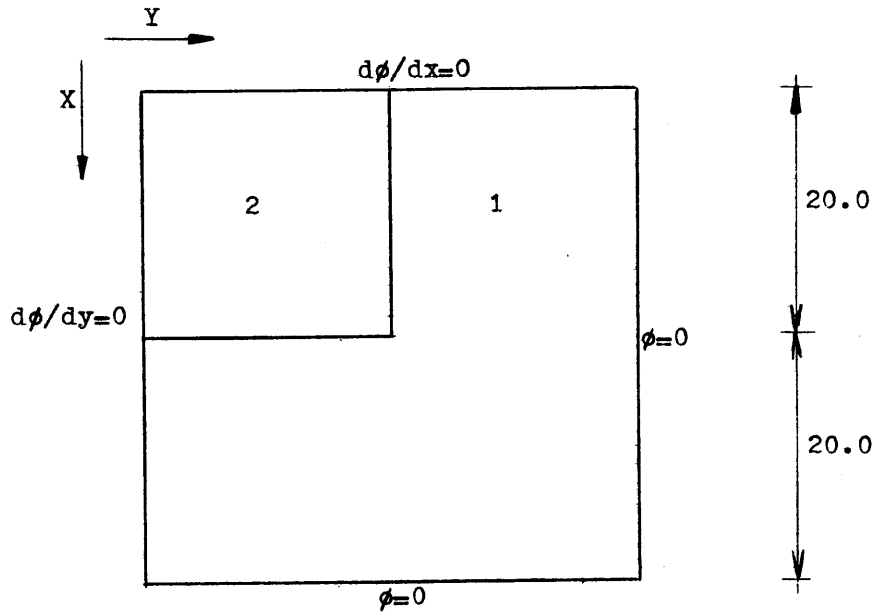
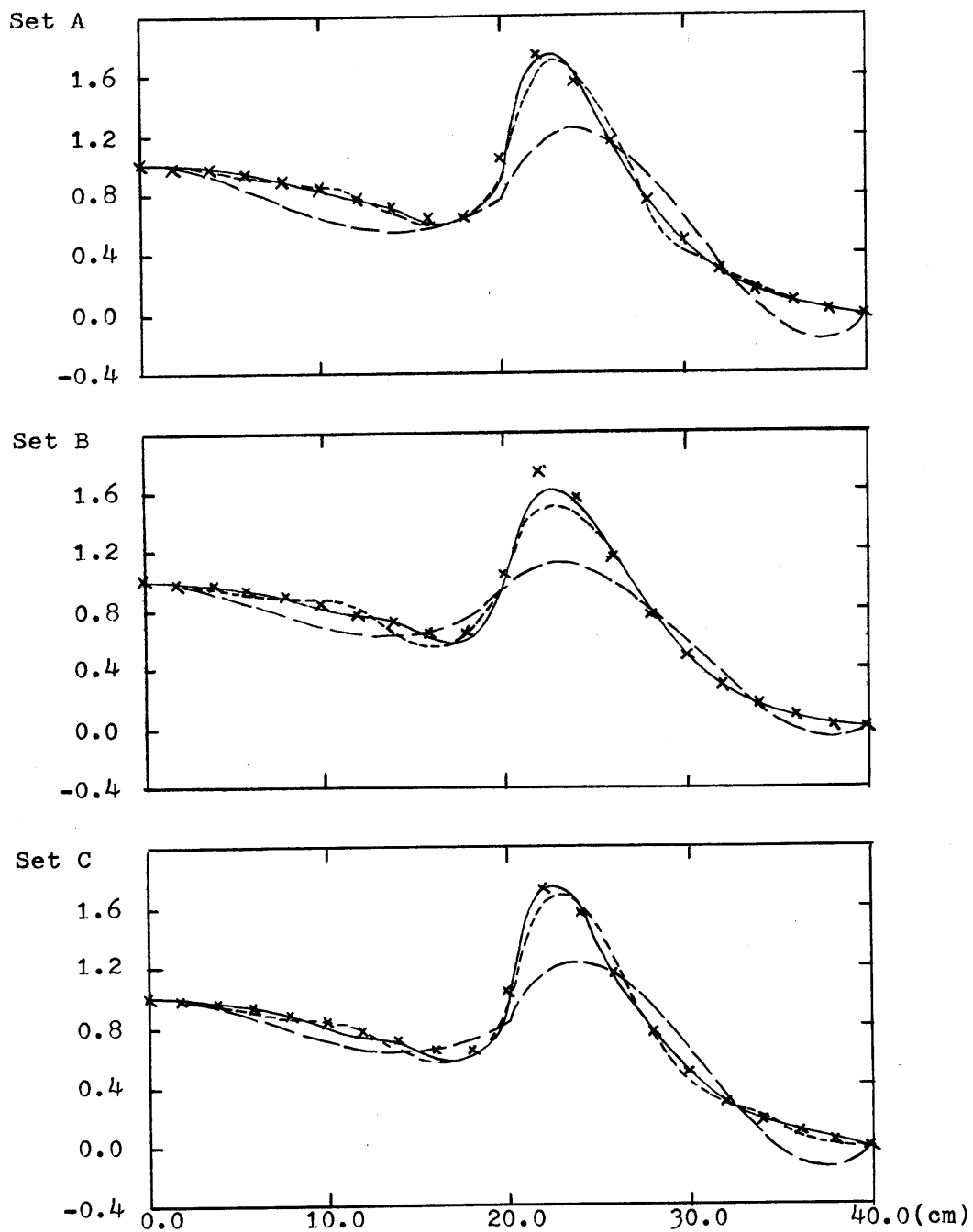


Table 3.4 - Eigenvalues for the basic two-dimensional problem.

Δh (cm)	Bicubic expansion			Finite difference
	Set A	Set B	Set C	
20.0	0.8962196	0.9041992	0.9023381	0.9004928
10.0	0.8965491	0.9008425	0.8980759	
6.667	0.8968995	0.8999750	0.8975901	
2.0				



KEY: Bicubic Hermite: --- $\Delta x=20.0$; - · - $\Delta x=13.333$; — $\Delta x=6.667$;
 Finite difference: $\Delta x=2.0$

Fig. 3.7.a - Thermal flux plots for basic two-dimensional problem at $x=0.0$

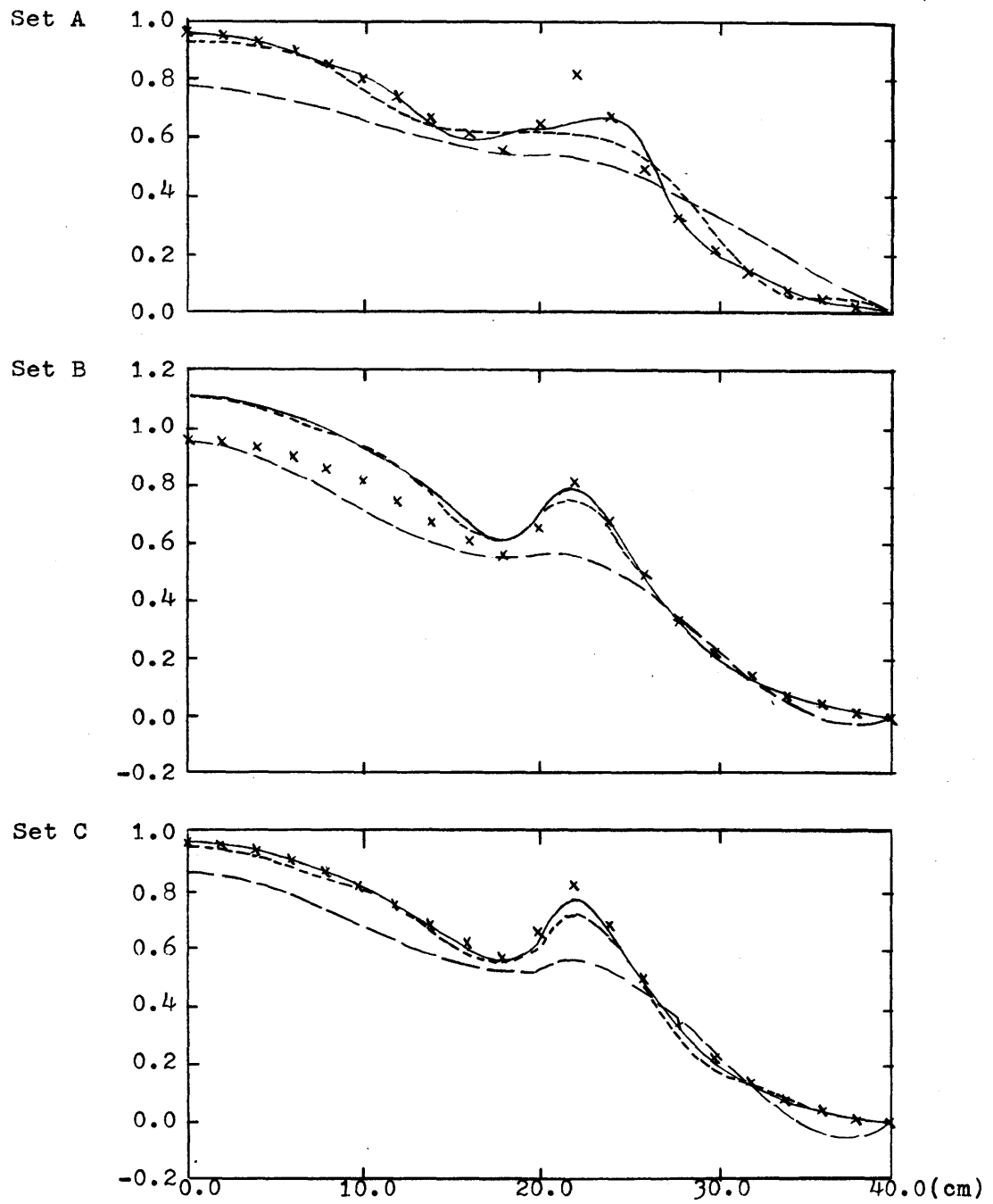


FIG. 3.7.b - Thermal flux plots for basic two-dimensional problem at $x=20.0$

compares very well with the finite difference one. For a 10 cm mesh, however, set C predicts very accurate pointwise results.

3.4 The basic two-dimensional problem with baffle surrounding the fuel region

The basic two-dimensional two-composition problem shown in Sec. (3.3) was first modified by the introduction of a SS baffle between fuel and reflector. Comparisons were then made between results achieved by treating the baffle explicitly (that is, by defining it explicitly as an expansion element) and by treating it implicitly (that is, by taking it as part of an overall baffle reflector expansion element where the cross sections are allowed to vary spatially).

The geometry used is shown in Fig. 3.8.

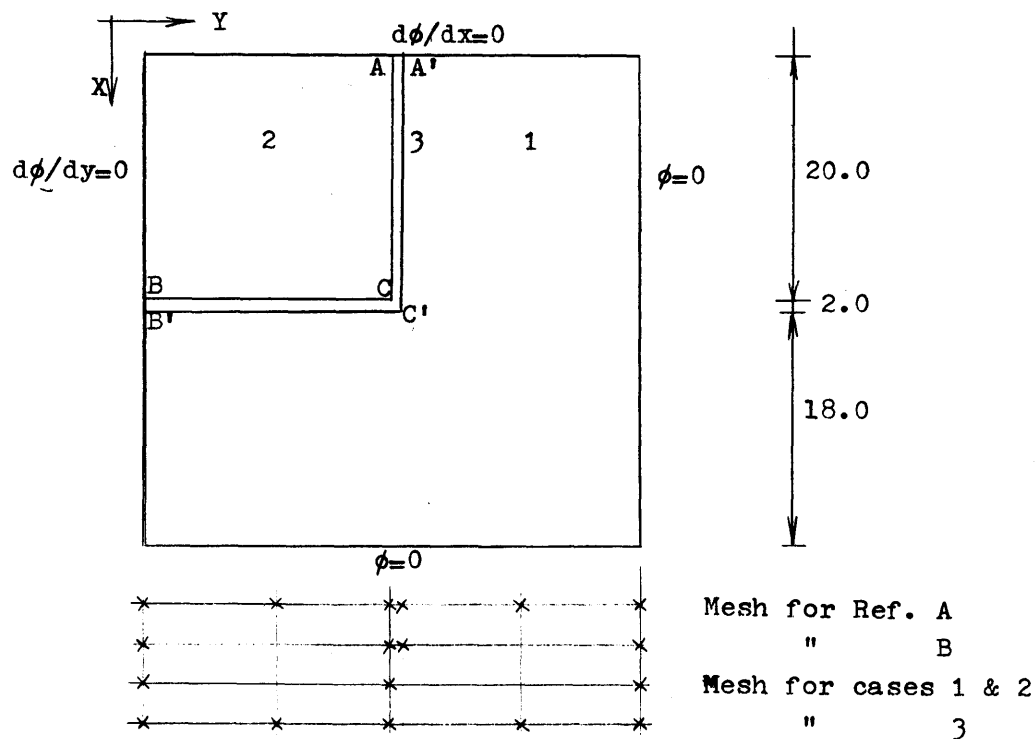


Fig. 3.8 - Geometry for the basic two-dimensional problem with baffle surrounding the fuel region.

For case 1 the set of functions was defined so as to explicitly preserve current between fuel and SS across points A and B (a set of 421 and a set of 422 type functions, consequently) whereas at the singular point the first derivatives were uncoupled (a 61-type set).

For cases 2 and 3 the first derivatives were uncoupled in the Y and X directions, respectively at points A and B (two 51-type sets, consequently).

Reference solutions A and B were obtained by treating the baffle region explicitly. Ref. A is the most accurate one and was obtained by uncoupling the first derivatives at all singular points (C and C'). Ref. B was obtained with a coarser mesh but with the same treatment for singular points C and C'.

Some results are presented in Table 3.6 and fluxes are plotted in Figs. 3.9.

Table 3.6 - Overall results for the basic two-dimensional problem with baffle surrounding the fuel region.

Case	Ref. A	Ref. B	11	2	3
k_{eff}	0.88385	0.88395	0.87931	0.88042	0.88142
# unknowns (/group)	104	40	18	20	70
Abs.rate (baffle + water)	17.4275	17.09072	18.0478	17.7249	17.7682
Fast/Thermal ratio (avg. fuel fluxes)	3.39073	3.38603	3.38638	3.40373	3.40571
Rel. running time	45			5	

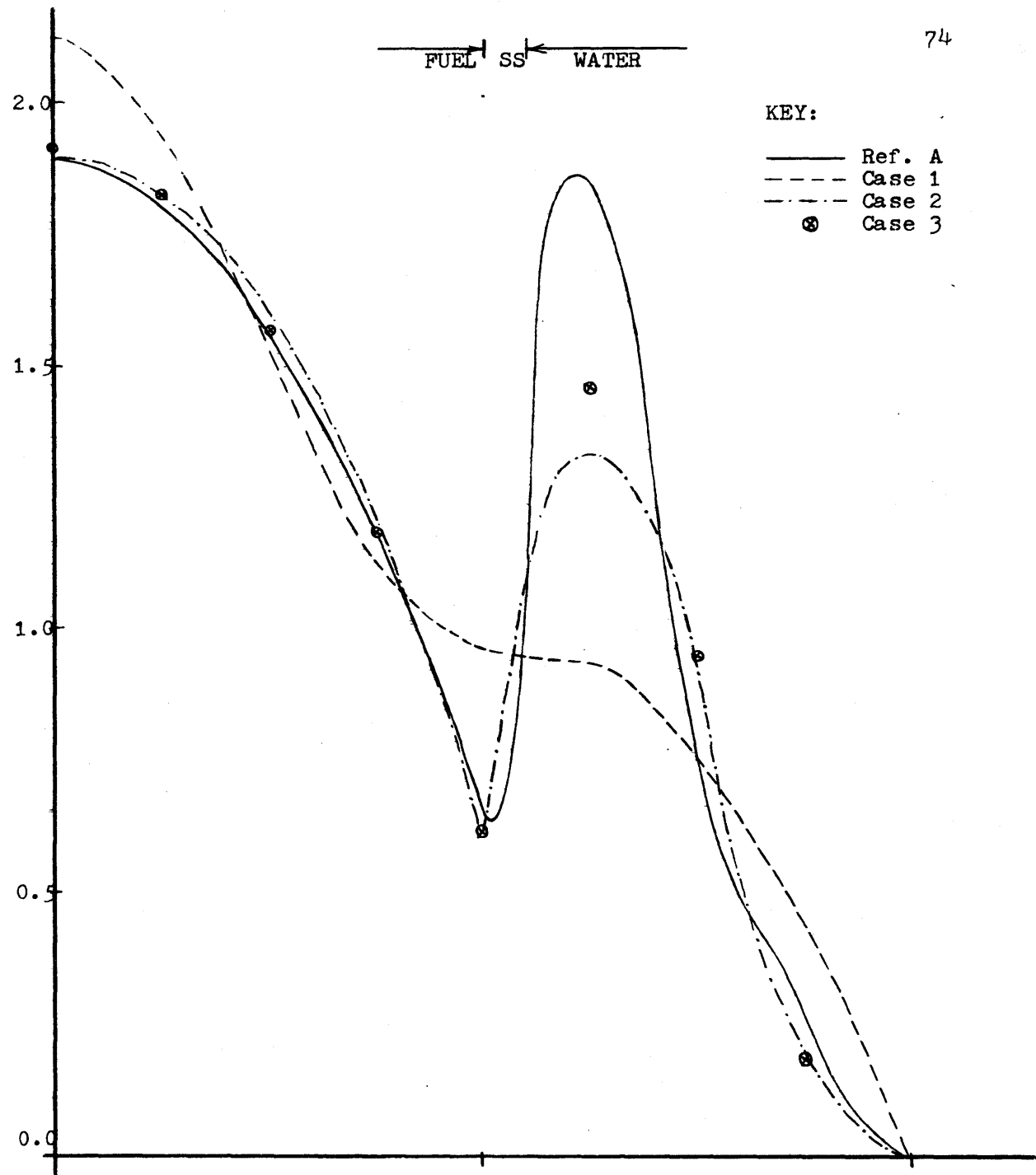


Fig. 3.9.a - Thermal fluxes for basic two-dimensional problem with baffle surrounding fuel region (y=0.0)

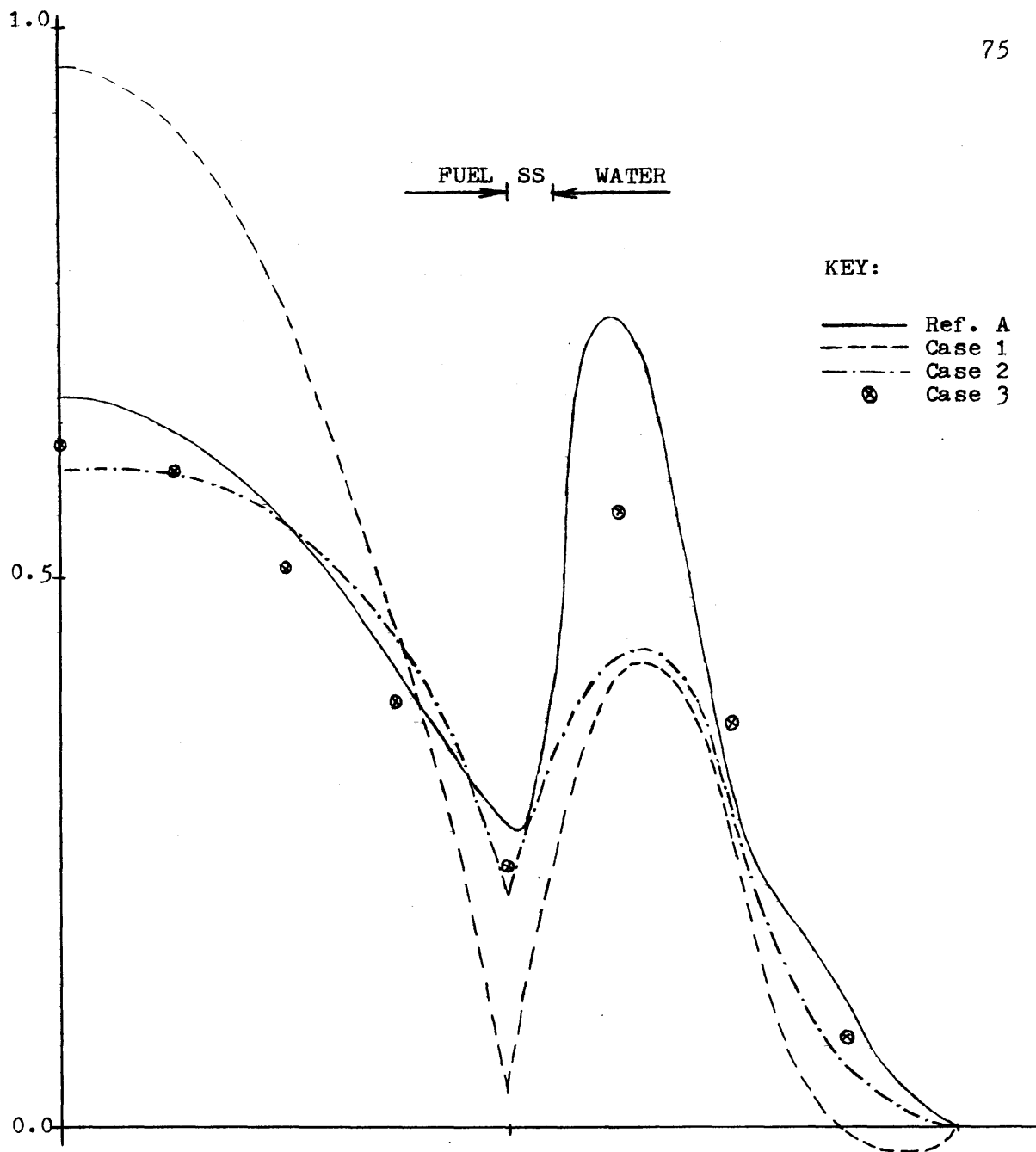


Fig. 3.9.b - Thermal fluxes for basic two-dimensional problem with baffle surrounding fuel region ($y=20.0$)

In Table 3.6 we can see that the effect of the baffle on the k_{eff} of the system (a reduction from about 0.9 to about 0.884) can be taken into account approximately through the implicit treatment of the baffle, although with an error of about 0.3% in the best case. It should be realized however that in a real reactor the influence of the baffle would not be as important as in the present case, so that the error might be acceptable.

Through the plots in Figs. 3.9 we clearly see how, in case 1, the method preserves current continuity at points A and B, this fact leading to distortions in the flux. At the singular point, however, where we do not try to preserve current, the solutions try to accommodate themselves. A logical extension then is to uncouple the derivatives in the Y direction at point A, and in the X direction at point B, because as we cannot represent the flux behavior with sufficient accuracy through baffle and water anyway there is no reason in preserving current at fuel-ss interfaces. This is done in cases 2 and 3: case 2 proved to be very accurate (within about 10% at interface points) for the fuel region, but less accurate at the reflector-baffle region; case 3 improved the plot a little bit, especially in the reflector. However, since we are not interested in the detailed flux behavior inside baffle and reflector but rather in their effect on fuel, we will probably get sufficient accuracy by using the case 2 setup.

It should also be realized that case 2 corresponds to a very coarse mesh and is probably intrinsically off a correct solution in the same measure as Ref. B is off Ref. A.

Fast flux plots are not shown but are very accurate in all cases.

3.5) The basic two-dimensional problem with varying cross sections inside the fuel region

The basic two-dimensional, two-composition problem shown in Sec. (3.3) is now modified by changing cross sections inside the fuel region. Comparisons will be made between results obtained by treating each sub-composition as a separate expansion element, and results obtained by treating the variations implicitly, that is by allowing them to occur inside coarse expansion elements. The results should give an idea about whether it is possible (or not) to define expansion elements over more than one fuel element in a real reactor.

The geometry used is shown in Fig. 3.10. The cross sections were changed in concentric "rings": the fast and thermal diffusion coefficients were increased by 10% whereas the thermal absorption cross section was almost doubled, from the outermost "ring" toward the inside (see Appendix A).

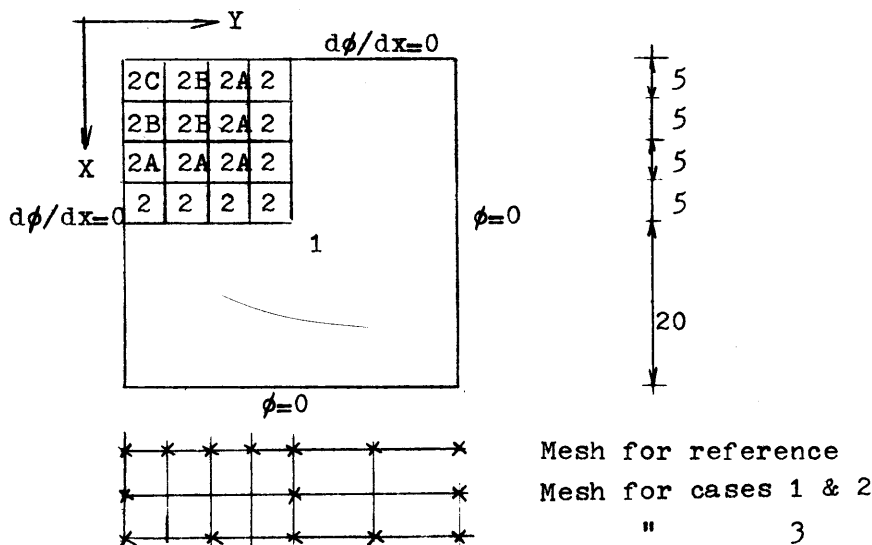


Fig. 3.10 - Geometry for the basic two-dimensional problem with varying cross sections in side the fuel region.

In case 1 we explicitly preserved current across points A and B, and in cases 2 and 3 this requirement is relaxed by uncoupling the derivatives in the Y direction (point A) and X direction (point B). In all the cases first derivatives were completely uncoupled at the singular point.

The reference solution was obtained with a very fine mesh, as shown. First derivatives were uncoupled at the fuel-water singular point, whereas derivative continuity was specified at all fuel-fuel singular points.

In Table 3.7 we shown some overall results, and in Figs. 3.11 and 3.12 fluxes are plotted.

Table 3.7 - Overall results for the basic two-dimensional problem with varying cross sections inside the fuel region.

Case	Reference	1	2(*)	3
k_{eff}	0.72936	0.74164	0.74271	0.72906
# unknowns(/group)	146	18	20	74
Abs. rate (fuel)	111.4182	109.94	109.98	111.423
Fast/thermal rat. (avg. fuel flux)	3.49786	3.346	3.3445	3.4912

(*) This case is presented as a sample problem in App.D

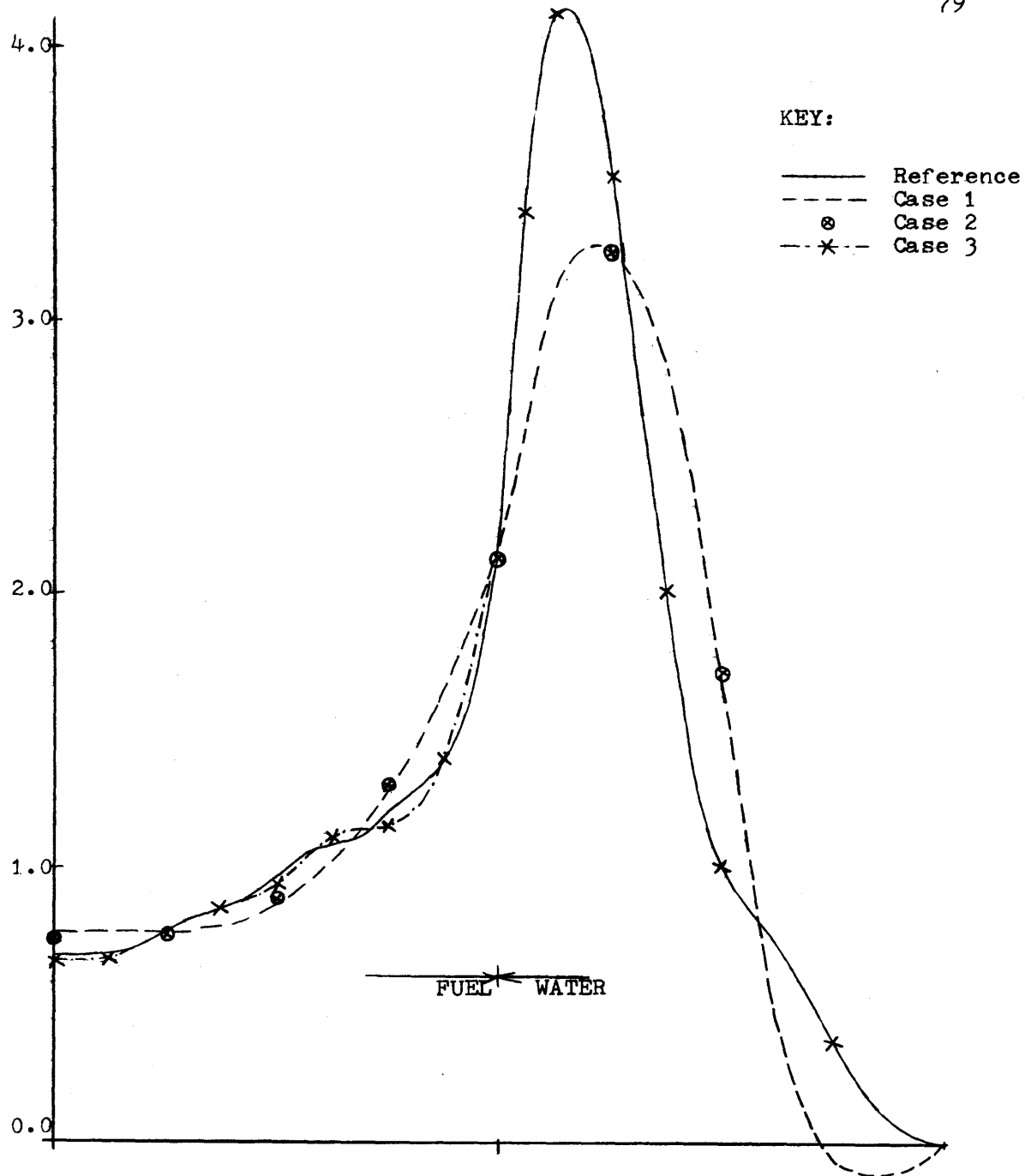


Fig. 3.11.a - Thermal fluxes for two-dimensional problem with varying cross sections inside fuel region ($x=0.0$)

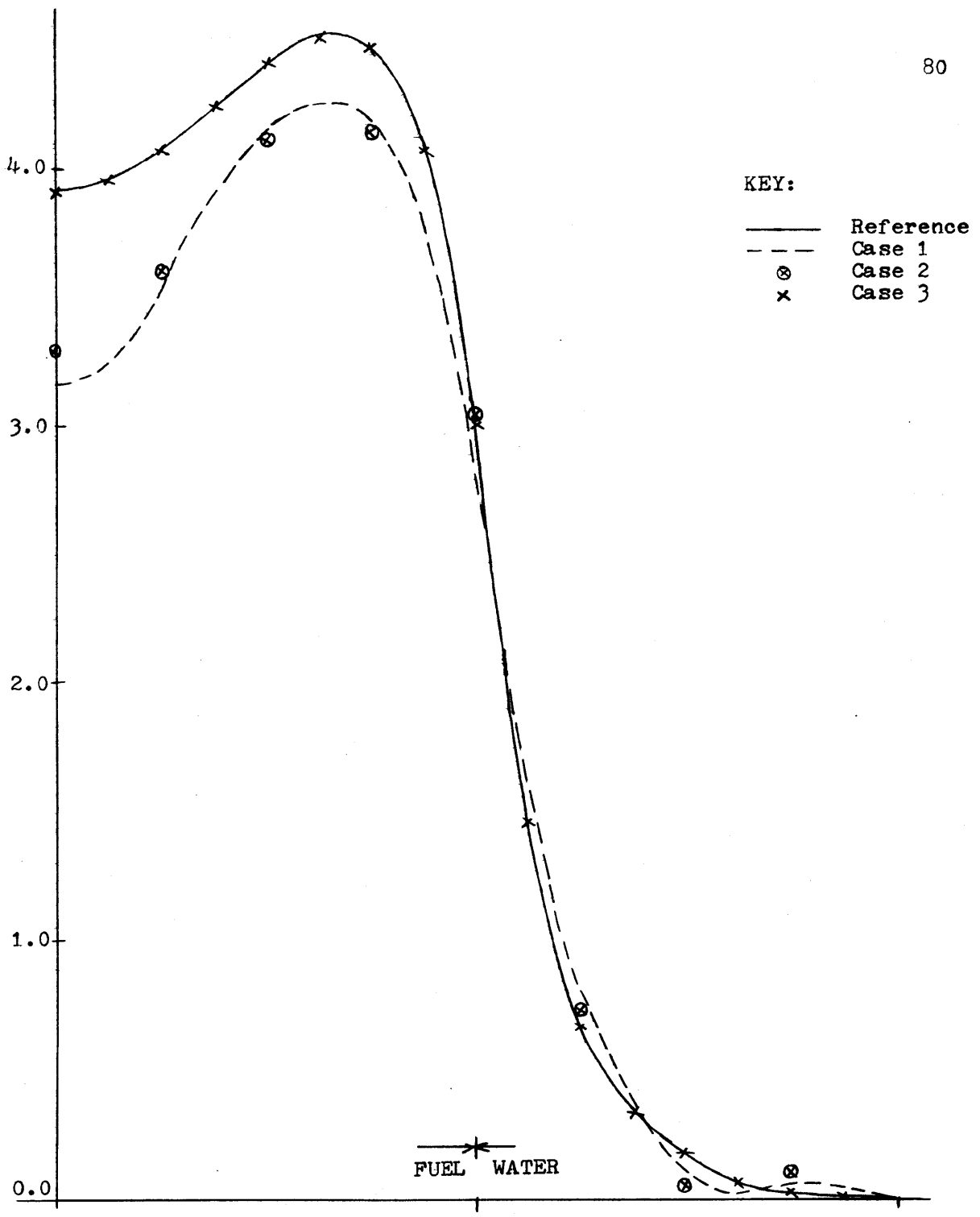


Fig. 3.11.b - Fast fluxes for two-dimensional problem with varying cross sections inside fuel region ($x=0.0$)

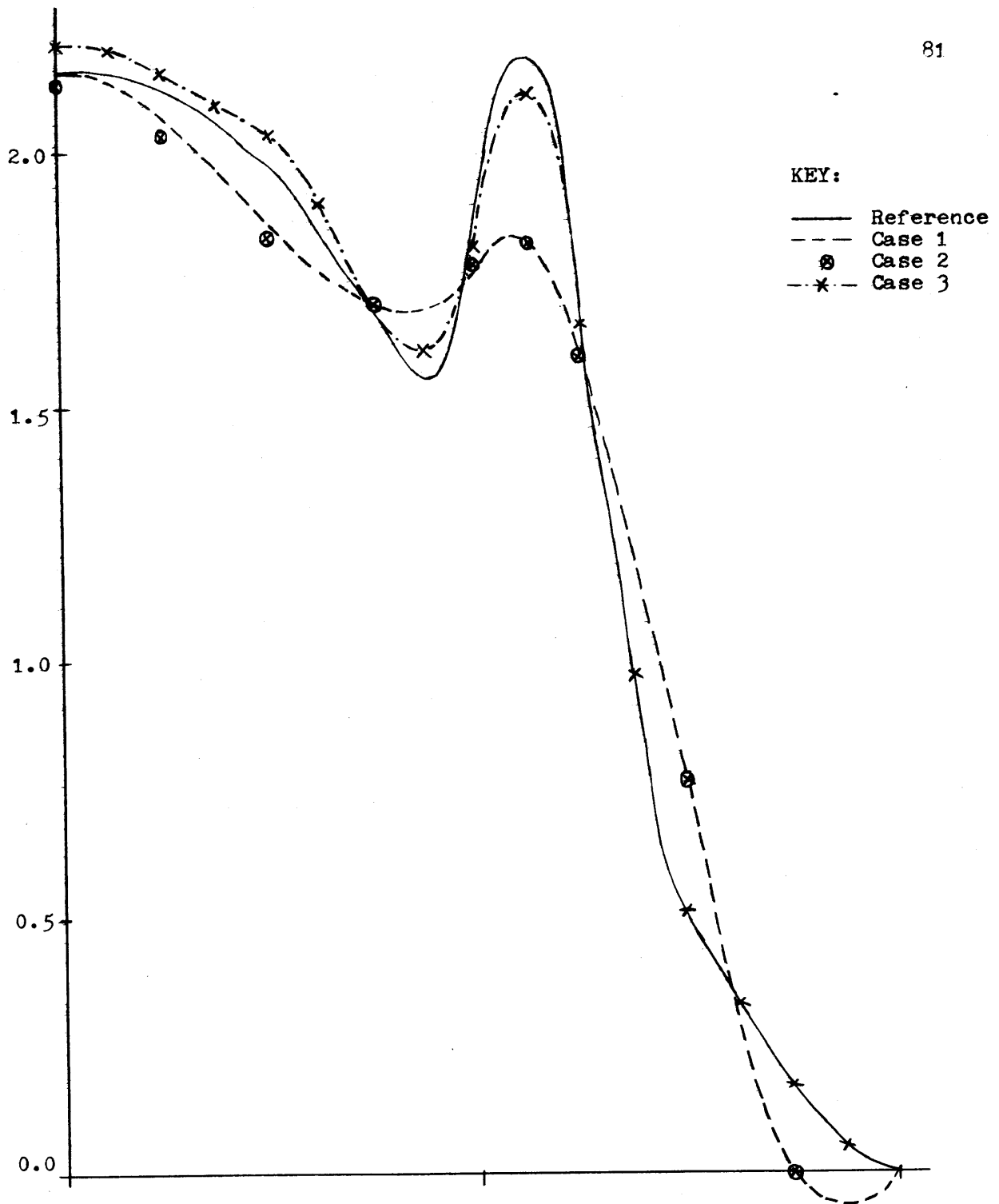


Fig. 3.12.a - Thermal fluxes for two-dimensional problem with varying cross sections inside fuel region ($x=20.0$)

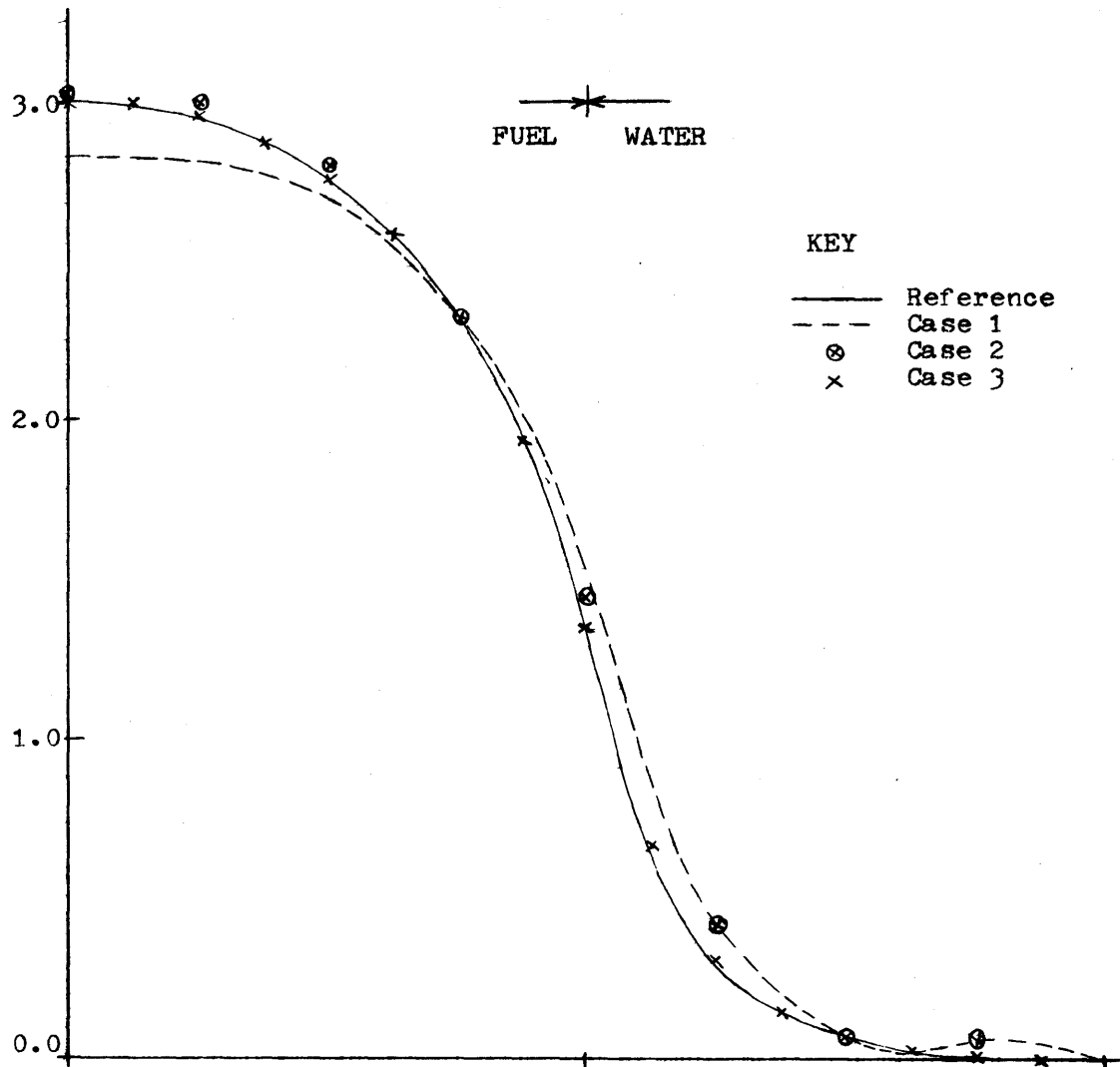


Fig. 3.12.b - Fast fluxes for two-dimensional problem with varying cross sections inside fuel region ($x=20.0$)

As it can be seen, cases 1 and 2 incorporate too coarse a mesh for the detailed flux behavior to be represented satisfactorily, whereas case 3 is probably an adequate setup for most applications: fast fluxes cannot be separated in the graphs, and thermal fluxes show perfectly the general behavior of the reference solution, even across singular points.

In these runs it is to be noted that it is not essential that the solution first derivatives be uncoupled across regular points such as A and B, a more important parameter being the number of subregions in the expansion element. This can be seen in the behavior of runs 1 and 2: whereas the number of unknowns is increased by one for each "opening", the results (mainly the overall ones) are very close. This is due to the fact that the flux solution is smoother in this case than in the baffle case shown in Sec. 3.4.

A series of runs was also made with this same geometry but with the diffusion coefficients kept constant throughout the region. The results were similar to the ones just shown and will not be presented here.

We can state now two conclusions from all the previous runs. First, when representing a real reactor we will probably be able to account adequately for the baffle's presence by treating it implicitly as part of an overall baffle-moderator expansion element. We should however uncouple appropriately the first derivatives of the solution at the baffle surface.

Second, we will probably be able to achieve reasonable results by defining one expansion element over four adjacent fuel elements. In this case, it is not essential that the first derivative be uncoupled. This conclusion is based upon the relatively small changes in D between regions.

We next proceed to show the results that were arrived at for the Zion 1 reactor in 3 different cases of region partition.

3.6) Results for Zion-1 in two-dimensions

Zion 1 is a modern PWR with a nominal capacity of 3250 Mw(th). The first cycle, quarter-core layout is shown in Fig. 3.13 (ref. 13). Once the fuel elements are homogenized, a quarter core, two-dimensional diffusion calculation may involve something between 2000 and 5000 unknowns per group or more, depending on the desired accuracy. In the present section, comparisons are made between CHD and PDQ-5 (ref. 2) and CITATION (ref. 15), for different mesh sizes.

PDQ-5 and CITATION are both standard diffusion-depletion codes. The first code considers the discrete points where flux is to be calculated as located at the borders of the finite difference volumes, whereas the second considers them located at the center of the volumes. This means that in the first code fluxes are calculated at material interfaces, but not so in the second code, since material properties have to be constant inside the finite difference volumes. This fact may cause the results to be slightly different for otherwise identical problems.

Three runs were made with CHD, the mesh spacing being as shown in Fig. 3.14.

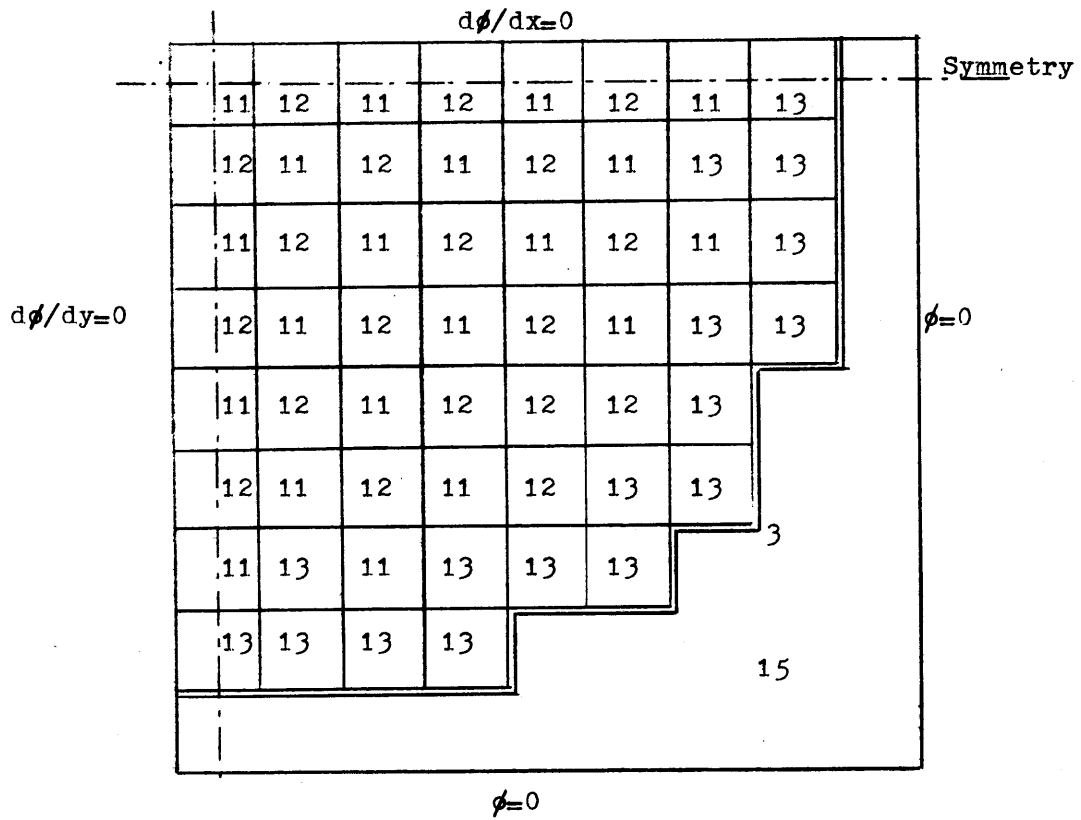


Fig. 3.13 - Zion 1 first core layout.

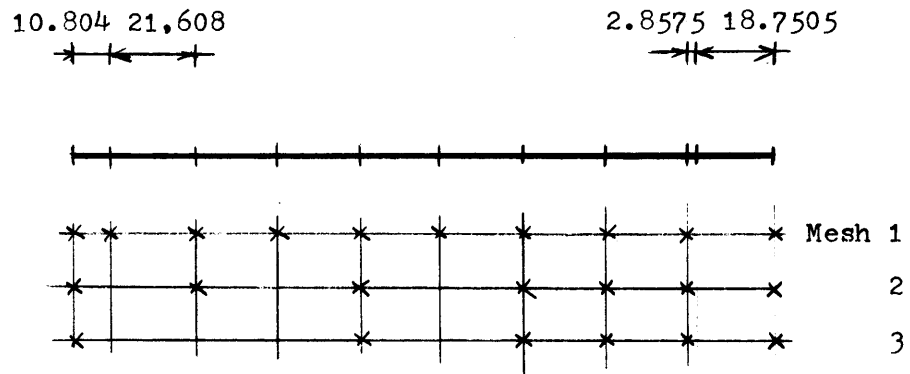


Fig. 3.14 - Mesh arrangement for CHD runs (Zion 1)

In all the runs the baffle was implicitly accounted for as a sub-region of an overall baffle-moderator expansion element. In these cases, as has been seen earlier, current continuity across the fuel-baffle interface is expected to introduce large distortions in the flux shape. We have, therefore, uncoupled the first derivatives of the solution at all these interface points (using sets 51 or 61, whichever is appropriate).

Run 1 was made by specifying set 41 (derivative continuity) for any singular point completely surrounded by fuel elements, and 61 (first derivatives uncoupling) otherwise.

For runs 2 and 3 first derivatives were uncoupled at all appropriate points.

Results are shown in Tables 3.8 and 3.9, and in Figs. 3.15 to 3.19.

We will first draw some conclusions from Table 3.8. All CHD runs do yield k_{eff} which are less than 0.04% off the most accurate finite difference value, in general with a number of unknowns between 5 and 10 times smaller. In particular we notice that CITATION (44 x 44) yields k_{eff} less accurately than CHD with the coarsest mesh. Under the computing environment at MIT we were able to run CHD with the finest mesh at a considerably lower cost as compared to PDQ-5, in spite of the fact that CHD does not incorporate any iteration acceleration technique.

Power ratios are shown in Figs. 3.15-17, whereas in Table 3.9 some relative differences are presented. We realize first that CHD (10 x 10) is remarkably accurate when compared to PDQ-5, whereas it is up to 10% off at the corner elements when compared to CITATION (75 x 75). However, PDQ-5 is also off

Table 3.8 - Overall results for Zion 1 as evaluated through PDQ-5, CITATION and CHD with several mesh sizes.

Code	PDQ-5	CITAT.	CITAT.	CHD	CHD	CHD
# unknowns(/group (mesh))	1936 (44-44)	1936 (44-44)	5625 (75-75)	348 (10-10)	190 (7-7)	128 (6-6)
k_{eff}	1.2749	1.27313	1.27508	1.27469	1.27474	1.27554
Pointwise flux convergence (10^{-4})	4.0	4.0	4.0	1.7	1.76	1.0
Fast/thermal flux ratio (mid point)	3.835	3.922		3.820	3.820	4.16
Fast/thermal flux ratio (fuel elements facing baf.)	5.133			5.191	5.181	5.183
Relative running cost (MIT conditions)	51			36		

	1	2	3	4	5	6	7	8
1	1.6271 1.6361 1.6524	1.7590 1.8002 1.8108	1.5320 1.5387 1.5524	1.5519 1.5856 1.5918	1.2537 1.2560 1.2623	1.1587 1.1758 1.1742	0.8039 0.7904 0.7896	0.5115 0.4954 0.4871
2		1.5800 1.5873 1.6025	1.6563 1.6051 1.7038	1.3945 1.3990 1.4087	1.3556 1.3818 1.3841	1.0372 1.0318 1.0339	0.9166 0.9230 0.9186	0.4970 0.4786 0.4705
3			1.4452 1.4490 1.4603	1.4664 1.4966 1.5023	1.1831 1.1813 1.1859	1.0779 1.0903 1.0878	0.7262 0.7092 0.7078	0.4455 0.4285 0.4206
4				1.2447 1.2439 1.2493	1.2123 1.2255 1.2257	0.9001 0.8872 0.8869	0.7216 0.7123 0.7067	0.3233 0.3013 0.2911
5					1.0777 1.0749 1.0747	0.8526 0.8401 0.8379	0.5340 0.5104 0.5019	
6						0.6682 0.6516 0.6462	0.3277 0.3042 0.2947	
7								
8								

Fig. 3.15 - Region averaged to core averaged power ratios.

KEY

CITAT.	(75 x 75)
PDQ-5	(44 x 44)
CHD	(10 x 10)

ZION 1

	1	2	3	4	5
1	1.6873 1.7069	1.5521 1.5656	1.2098 1.2138	0.8788 0.8775	0.4842 0.4756
2		1.4215 1.4300	1.0961 1.0949	0.7108 0.7032	0.3649 0.3527
3			0.8517 0.8468	0.4073 0.3961	
4					
5					

Fig. 3.16 - Region averaged to core averaged power ratios.

PDQ-5	(44 x 44)
CHD	(7 x 7)

	1	2	3	4
1	1.5343 1.5480	1.1448 1.1459	0.7828 0.7824	0.4160 0.4057
2		0.8517 0.8408	0.4073 0.3911	
3				
4				

Fig. 3.17 - Region averaged to core averaged power ratios.

PDQ-5	(44 x 44)
CHD	(6 x 6)

Table 3.9 - Relative differences of average power in expansion elements of Zion 1 as evaluated through CHD relative to PDQ-5 and CITATION.

Position (see Figs. 3.15 through 3.17)	CHD mesh	CHD off relative to (%)	
		PDQ-5 (44-44)	CITAT.(75-75)
1-1 (3.15)	10-10	+ 0.996	+ 1.555
1-1 (3.16)	7-7	+ 1.162	
1-1 (3.17)	6-6	+ 0.893	
1-8 (3.15)	10-10	- 1.675	- 4.770
1.5 (3.16)	7-7	- 1.776	
1.4 (3.17)	6-6	- 2.476	
6-7 (3.15)	10-10	- 3.123	- 10.070
3-4 (3.16)	7-7	- 2.750	
2-3 (3.17)	6-6	- 3.977	

Fig. 3.18 - Flux plots for Zion 1
as evaluated through PDQ-5 (44 x 44)
and CHD (10 x 10)

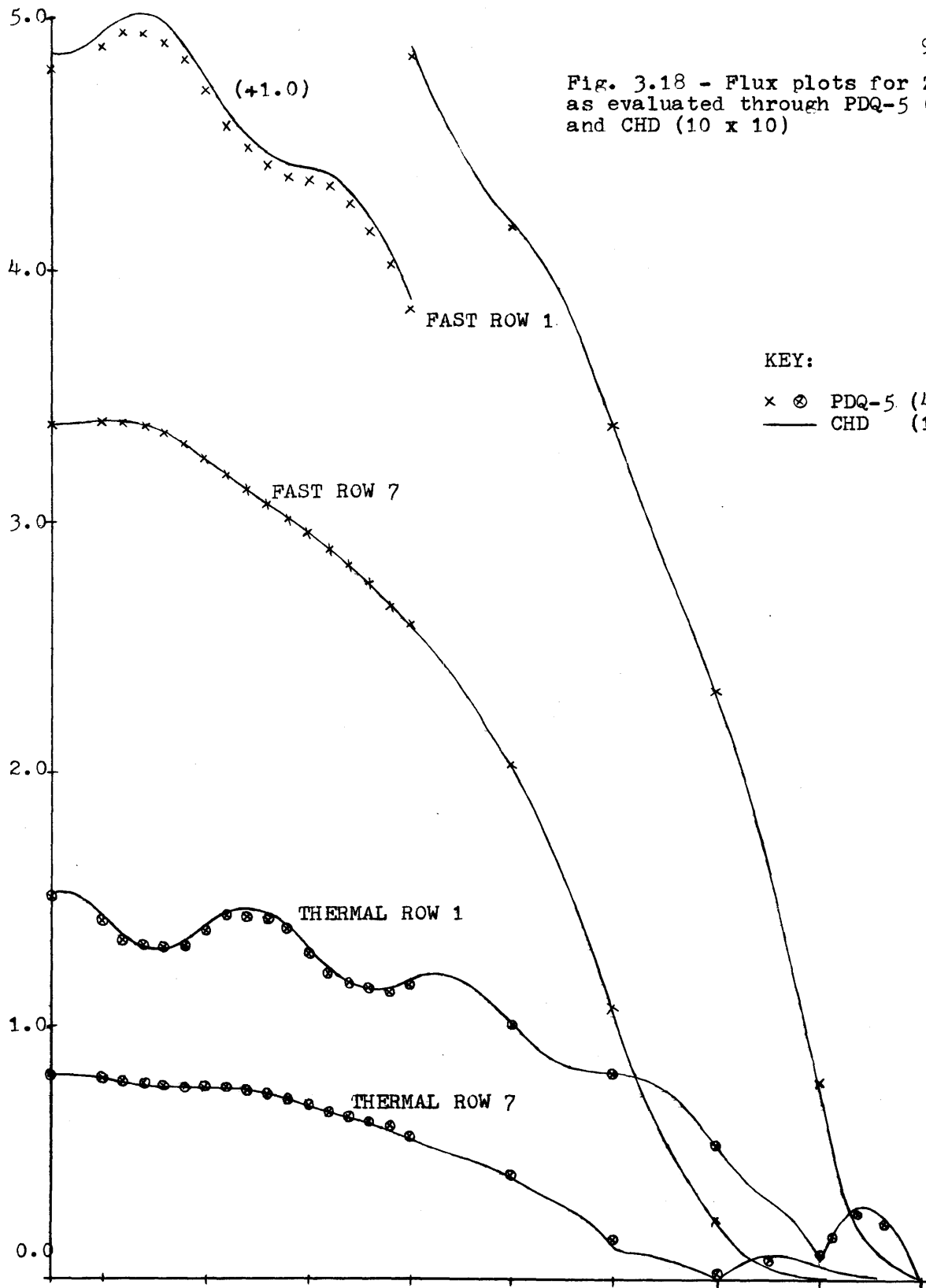
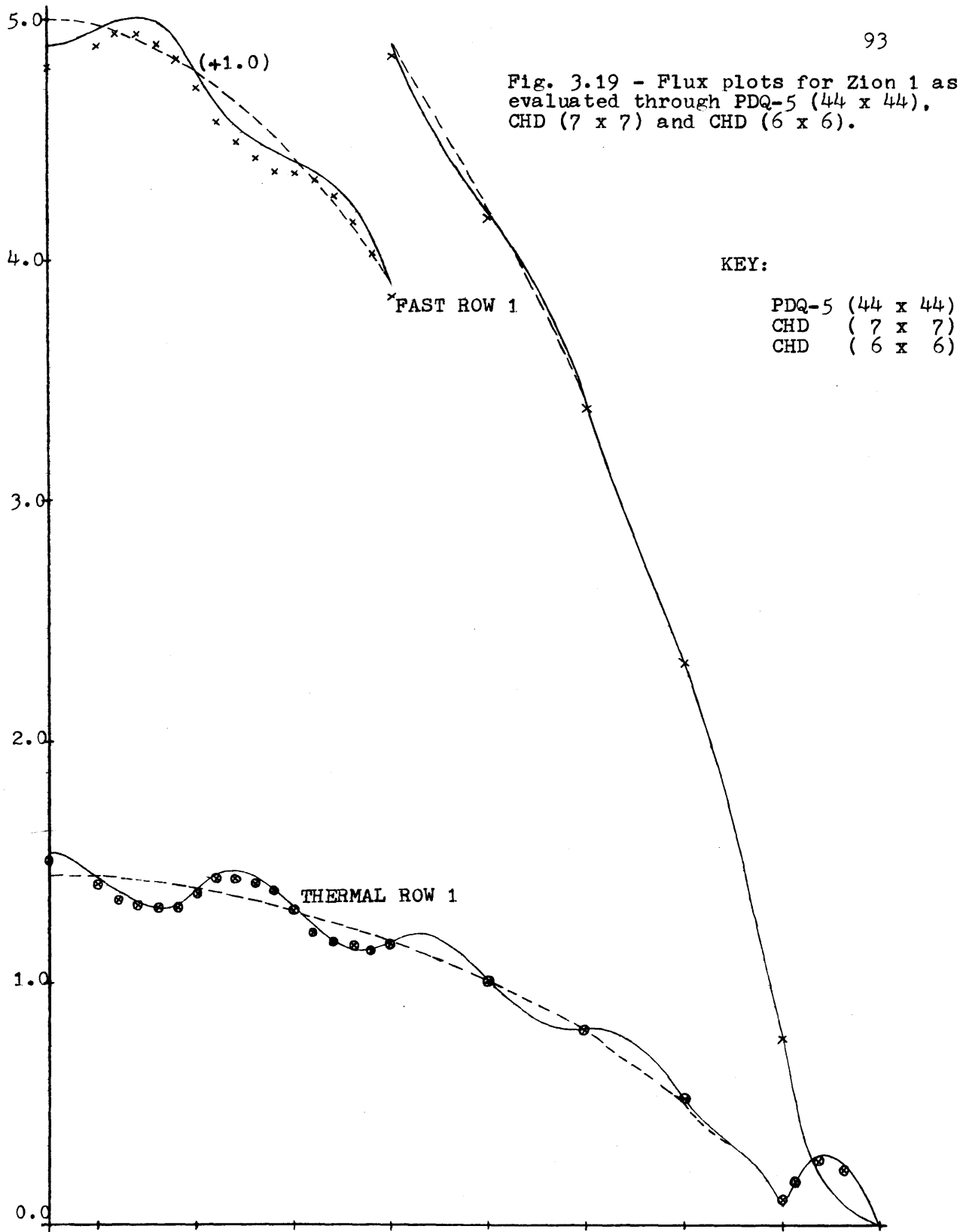


Fig. 3.19 - Flux plots for Zion 1 as evaluated through PDQ-5 (44 x 44), CHD (7 x 7) and CHD (6 x 6).



CITATION by about 7% in the worst case, so that we are lead to conclude that, very roughly, CHD (10 x 10) and PDQ-5 (44 x 44) show the same order of accuracy.

The coarser meshes CHD runs show larger differences when compared to PDQ-5. However, the very largest mesh CHD set up might provide basis for the development of reactor simulators of the FLARE type, without the inconveniences of this code.

Flux shapes are shown in Figs. 3.18 and 3.19. It is clear that the shape yielded by the smallest mesh CHD run is essentially correct with differences of the same order as those shown in Table 3.9 for the power ratios. The shape detail achieved by CHD (7 x 7) where expansion elements are defined over up to 4 fuel elements are remarkably accurate. However, when the mesh is opened still more for CHD (6 x 6) we are no longer able to follow the fine spatial oscillations of the solution.

CHAPTER IV

CONCLUSIONS AND RECOMMENDATIONS4.1) Conclusions

From the results presented in Ch. III we draw several general conclusions. We have demonstrated that it is possible to represent a multi-region reactor, in two-dimensional cartesian geometry, with a fair accuracy, using Hermite bicubic polynomials expansion on a coarse mesh partition equivalent to a PWR fuel element. The resultant system of equations requires at least 5 times fewer unknowns, for equivalent accuracy, than a system based on the finite difference method, and thus offers the potential of faster codes. These conclusions are drawn from Sec. 3.1 where EXTERMINATOR 2 is compared with CHD for a 4-composition problem, and from Sec. 3.6 (case 1) where PDQ-5 and CITATION are compared with CHD for the Zion 1 reactor. In particular for this latter case we state the following numerical results:

a) CHD yields k_{eff} which is 0.03% off a value obtained with a very fine finite difference mesh (CITATION) using 15 times more unknowns.

b) CHD yields fuel element averaged fission rates which are 1% and 3.2% (for the innermost and an outside element, respectively) off the values predicted by PDQ-5 using 6 times more unknowns.

We have further shown that it is possible to define expansion elements over regions with varying cross sections, by implicitly accounting for the variation in the construction of the coefficient matrices. This conclusion is drawn from Secs. 3.4 and 3.5, where

some convenient small two-dimensional problems are investigated, and from Sec. 3.6 (cases 1, 2 and 3), where PDQ-5 and CITATION results are compared to coarse mesh CHD results for the Zion 1 reactor. In particular we state the following results.

a) The core baffle of a PWR can be accounted for by implicitly treating it as a sub-region of an overall expansion region incorporating baffle and the nearby reflector. The baffle's influence on the core can be well accounted for in this way, the pointwise flux distribution in the nearby fuel elements being remarkably accurate.

b) The definition of one expansion element over 4 fuel elements in some regions yields very accurate results both in pointwise flux distributions and in overall reactor characteristics. With a number of unknowns 10 times smaller CHD was able to follow very closely pointwise flux distributions obtained with PDQ-5; further, region-averaged fission rates were less than 3% off and k_{eff} was less than 0.03% off the most accurate value.

c) Very coarse meshes (up to 16 fuel elements) are feasible for PWR's. The results are reasonably accurate (k_{eff} less than 0.04% off the most accurate value) and the number of unknowns is remarkably small (128 per group for the Zion 1 reactor on a quarter core symmetry).

In all cases the differences in material properties were close to what one would find in real reactors. In particular we have used actual Zion 1, homogenized cross sections (see App. A) whenever the Zion 1 reactor was represented.

All these results seem to indicate that it is possible to

build a few-group, three-dimensional, reflected reactor simulator of the FLARE type without any of the inconveniences of present nodal methods.

4.2) Recommendations

Problems that should be investigated following this thesis are:

- a) Acceleration of convergence through Chebyshev polynomials.
- b) Desirability and feasibility of using different mesh sizes for the thermal and fast groups.
- c) Mesh lines that do not extend over the whole problem domain (call it virtual mesh lines and virtual mesh points). Assume a mesh as shown in Fig. 4.1.

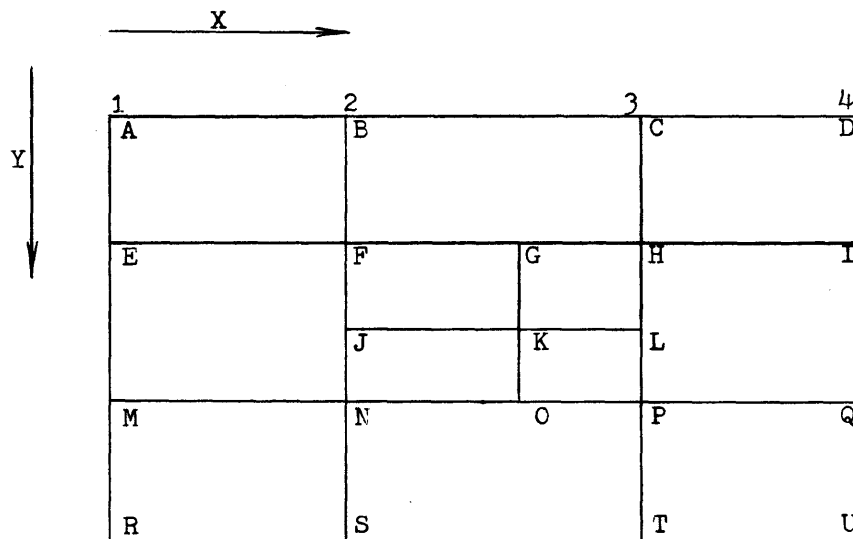


Fig. 4.1 - Regular and virtual mesh lines and points in two dimensions

Points (11), (21), ..., (44) are part of a regular mesh. The virtual point K can be incorporated into the mesh structure without extending lines JL and GO throughout the domain. To achieve this we define the following expansion functions:

<u>Point</u>	<u>Interval of definition</u>
N	RSTPOKJFEMR
J	FGKONJF
F	AEMNJKGHCB
K	NOPLHGFJN

If we want to preserve continuities across point J (for instance) we assume the corresponding unknowns to be determined by conditions at N and F. In other words, we make

$$\begin{aligned}\phi(J_x^-) &= \phi(J_x^+) \\ \partial\phi/\partial x(J^-) &= \partial\phi/\partial x(J^+)\end{aligned}$$

d) Zero flux boundaries which are not straight lines but which follow the core shape.

e) Determination of the maximum size of matrices that can be handled with a Cholesky factorization for normal problems.

f) When these questions are answered it will probably be worthwhile to develop a coarse mesh reactor simulator using the general setup shown in Sec. 3.6 (case 2 or 3), still with a direct inversion in the in-group solutions.

REFERENCES

- 1) A. F. Henry; Nuclear Reactor Physics II, MIT Lecture Notes; Spring, 1971.
- 2) L. A. Hageman, C. J. Pfeifer; The Utilization of the Neutron Diffusion Program PDQ-5; WAPD-TM-395; January, 1965.
- 3) W. R. Cadwell; PDQ-7 Reference Manual; WAPD-TM-678; January, 1967
- 4) Richard S. Varga; Matrix Iterative Analysis, Sec. 6.3; Prentice-Hall inc.; 1962.
- 5) D. L. Delp, et al.; FLARE - A Three - Dimensional Boiling Water Simulator; GEAP-4598; July, 1964.
- 6) K. F. Hansen, C. M. Kang; Finite Element Methods in Reactor Physics Analysis; to be published in Advances in Nuclear Science and Technology.
- 7) Chang Mu Kang, K. F. Hansen; Finite Element Methods for Space - Time Reactor Analysis; U.S.AEC Report MIT - 3903 - 5, November, 1971.

Also appears as:

Chang Mu Kang; ScD Thesis, Nuclear Engineering Department, MIT; November, 1971.

- 8) H. G. Kaper, G. K. Leaf, A. J. Lindeman; Applications of Finite Element Methods in Reactor Mathematics. Numerical Solution of the Neutron Diffusion Equation; ANL-7925; February, 1972.
- 9) L. A. Semenza, E. E. Lewis, E. C. Rossow; The Application of the Finite Element Method to the Multigroup Neutron Diffusion Equation; Nucl. Sci. Eng., 47, 302-310 (1972).
- 10) Patrick G. Bailey, A. F. Henry; Variational Derivation of Modal-Nodal Finite Difference Equations in Spatial Reactor Physics; U.S.AEC Report COO-3052-5; July, 1972.
- 11) L. A. Hageman; Numerical Methods and Techniques Used in the Two-Dimensional Neutron Diffusion Program PDQ-5; WAPD-TM-364; February, 1963.

- 12) V. N. Faddeeva; Computational Methods of Linear Algebra; Dower Publications, Inc.; New York; 1959.
- 13) Terrance A. Rieck; The Engineering Feasibility and Fuel Cost Analysis of Variable Nuclear Refueling Intervals; PhD Thesis in progress, Nuclear Engineering Dpt., MIT.
- 14) T. B. Fowler, M. L. Tobias, D. R. Vondy; EXTERMINATOR 2 - a Fortran IV Code for Solving Multigroup Neutron Diffusion Equations in Two Dimensions; ORNL 4078; April, 1967.
- 15) T. B. Fowler, D. R. Vondy, G. W. Cunningham; Nuclear Reactor Core Analysis Code: CITATION; ORNL-TM-2496, Rev. 2; July, 1969.

MACROSCOPIC CROSS SECTIONSGroup 1
Group 2

Compo- sition	D(cm)	$\Sigma_r(\text{cm}^{-1})$	$\nu\Sigma_f(\text{cm}^{-1})$	$\Sigma_s(\text{cm}^{-1})$ ($k \rightarrow k+1$)	Comments
1	1.20 0.15	0.101 0.020	0.0 0.0	0.1	C.M.Kang (water)
2	1.5 0.4	0.0623 0.2	0.0 0.218	0.06	C.M.Kang (fuel).
2A	1.55 0.4133	0.0623 0.25	0.0 0.218	0.06	
2B	1.60 0.4267	0.0623 0.30	0.0 0.218	0.06	
2C	1.65 0.44	0.0623 0.35	0.0 0.218	0.06	
3	1.02130 0.33548	0.00322 0.14596	0.0 0.0	0.0	Zion 1 (core baffle)
5	1.63 0.275	0.0383 0.0108	0.0 0.0	0.0380	Water
6	1.399 0.387	0.0261 0.0832	0.00659 0.129	0.0168	Fuel (high enrich.)
7	1.397 0.389	0.0260 0.071	0.00553 0.102	0.0172	Fuel (medium enrich.)
8	1.397 0.388	0.0259 0.0532	0.00485 0.0636	0.0179	Fuel (low enrich.)
11	1.41760 0.37335	0.02597 0.06669	0.00536 0.10433	0.01742	Zion 1 (2.25 %)
12	1.41970 0.37370	0.02576 0.07606	0.00601 0.12472	0.01694	Zion 1 (2.8 %)
13	1.42650 0.37424	0.02560 0.08359	0.00653 0.14120	0.01658	Zion 1 (3.3 %)
14	1.38377 0.29745	0.025164 0.032006	0.0 0.0	0.024241	16.5 % SS + 83.5 % water
15	1.45540 0.28994	0.02950 0.00949	0.0 0.0	0.02903	Zion 1 (water)

APPENDIX B
INNER PRODUCTS BETWEEN FINE LIMITS
FOR CUBIC HERMITE POLYNOMIALS

$$h_- \equiv x_i - x_{i-1}$$

$$h_+ \equiv x_{i+1} - x_i$$

$$x_2 > x_1$$

$$(x)^k \equiv (x_2^k - x_1^k)$$

$$(h-x)^k \equiv [(h-x_2)^k - (h-x_1)^k]$$

$$\begin{aligned} (u_1^{0-}, u_{i-1}^{0+})_{x_1}^{x_2} &\equiv \int_{x_1}^{x_2} u_1^{0-}(x) u_{i-1}^{0+}(x) dx = \\ &= -\frac{4}{7} h_-^{-6} (x)^7 + 2h_-^{-5} (x)^6 - \frac{9}{5} h_-^{-4} (x)^5 \\ &\quad - \frac{1}{2} h_-^{-3} (x)^4 + h_-^{-2} (x)^3 \end{aligned}$$

$$(u_1^{0-}, u_i^{0-})_{x_1}^{x_2} = \frac{4}{7} h_-^{-6} (x)^7 - 2h_-^{-5} (x)^6 + \frac{9}{5} h_-^{-4} (x)^5$$

$$\begin{aligned} (u_1^{0+}, u_{i+1}^{0-})_{x_1}^{x_2} &= -\frac{4}{7} h_+^{-6} (x)^7 + 2h_+^{-5} (x)^6 - \frac{9}{5} h_+^{-4} (x)^5 \\ &\quad - \frac{1}{2} h_+^{-3} (x)^4 + h_+^{-2} (x)^3 \end{aligned}$$

$$(u_1^{0+}, u_i^{0+})_{x_1}^{x_2} = -\frac{4}{7} h_+^{-6} (h_+-x)^7 + 2h_+^{-5} (h_+-x)^6 - \frac{9}{5} h_+^{-4} (h_+-x)^5$$

$$(u_i^{0-}, u_{i-1}^{1+})_{x_1}^{x_2} = -\frac{2}{7} h_-^{-5} (x)^7 + \frac{7}{6} h_-^{-4} (x)^6 - \frac{8}{5} h_-^{-3} (x)^5 + \frac{3}{4} h_-^{-2} (x)^4$$

$$(u_i^{1+}, u_{i+1}^{0-})_{x_1}^{x_2} = -\frac{2}{7} h_+^{-5} (x)^7 + \frac{7}{6} h_+^{-4} (x)^6 - \frac{8}{5} h_+^{-3} (x)^5 + \frac{3}{4} h_+^{-2} (x)^4$$

$$(u_i^{0+}, u_{i+1}^{1-})_{x_1}^{x_2} = \frac{2}{7} h_+^{-5} (x)^7 - \frac{5}{6} h_+^{-4} (x)^6 + \frac{3}{5} h_+^{-3} (x)^5 + \frac{1}{4} h_+^{-2} (x)^4 - \frac{1}{3} h_+^{-1} (x)^3$$

$$(u_i^{1-}, u_{i-1}^{0+})_{x_1}^{x_2} = \frac{2}{7} h_-^{-5} (x)^7 - \frac{5}{6} h_-^{-4} (x)^6 + \frac{3}{5} h_-^{-3} (x)^5 + \frac{1}{4} h_-^{-2} (x)^4 - \frac{1}{3} h_-^{-1} (x)^3$$

$$(u_i^{0-}, u_i^{1-})_{x_1}^{x_2} = -\frac{2}{7} h_-^{-5} (x)^7 + \frac{5}{6} h_-^{-4} (x)^6 - \frac{3}{5} h_-^{-3} (x)^5$$

$$(u_i^{1-}, u_i^{0-})_{x_1}^{x_2} = (u_i^{0-}, u_i^{1-})_{x_1}^{x_2}$$

$$(u_i^{0+}, u_i^{1+})_{x_1}^{x_2} = -\frac{2}{7} h_+^{-5} (h_+ - x)^7 + \frac{5}{6} h_+^{-4} (h_+ - x)^6 - \frac{3}{5} h_+^{-3} (h_+ - x)^5$$

$$(u_i^{1+}, u_i^{0+})_{x_1}^{x_2} = (u_i^{0+}, u_i^{1+})_{x_1}^{x_2}$$

$$(u_i^{1-}, u_{i-1}^{1+})_{x_1}^{x_2} = \frac{1}{7} h_-^{-4} (x)^7 - \frac{1}{2} h_-^{-3} (x)^6 + \frac{3}{5} h_-^{-2} (x)^5 - \frac{1}{4} h_-^{-1} (x)^4$$

$$(u_i^{1+}, u_{i+1}^{1-})_{x_1}^{x_2} = \frac{1}{7} h_+^{-4} (x)^7 - \frac{1}{2} h_+^{-3} (x)^6 + \frac{3}{5} h_+^{-2} (x)^5 - \frac{1}{4} h_+^{-1} (x)^4$$

$$(u_i^{1-}, u_i^{1-})_{x_1}^{x_2} = \frac{1}{7} h_-^{-4} (x)^7 - \frac{1}{3} h_-^{-3} (x)^6 + \frac{1}{5} h_-^{-2} (x)^5$$

$$(u_i^{1+}, u_i^{1+})_{x_1}^{x_2} = -\frac{1}{7} h_+^{-4} (h_+ - x)^7 + \frac{1}{3} h_+^{-3} (h_+ - x)^6 - \frac{1}{5} h_+^{-2} (h_+ - x)^5$$

$$\left(\frac{d}{dx} u_i^{0-}, \frac{d}{dx} u_{i-1}^{0+}\right)_{x_1}^{x_2} = \int_{x_1}^{x_2} \frac{d}{dx} u_i^{0-}(x) \frac{d}{dx} u_{i-1}^{0+}(x) dx =$$

$$= -\frac{36}{5} h_-^{-6} (x)^5 + 18 h_-^{-5} (x)^4 - 12 h_-^{-4} (x)^3$$

$$\left(\frac{d}{dx} u_i^{0+}, \frac{d}{dx} u_{i+1}^{0-}\right)_{x_1}^{x_2} = -\frac{36}{5} h_+^{-6} (x)^5 + 18 h_+^{-5} (x)^4 - 12 h_+^{-4} (x)^3$$

$$\left(\frac{d}{dx} u_i^{0-}, \frac{d}{dx} u_i^{0-}\right)_{x_1}^{x_2} = \frac{36}{5} h_-^{-6} (x)^5 - 18 h_-^{-5} (x)^4 + 12 h_-^{-4} (x)^3$$

$$\left(\frac{d}{dx} u_i^{0+}, \frac{d}{dx} u_i^{0+}\right)_{x_1}^{x_2} = -\frac{36}{5} h_+^{-6} (h-x)^5 + 18 h_+^{-5} (h-x)^4 - 12 h_+^{-4} (h-x)^3$$

$$\left(\frac{d}{dx} u_1^{1-}, \frac{d}{dx} u_{1-1}^{0+}\right)_{x_1}^{x_2} = \frac{18}{5} h_-^{-5} (x)^5 - \frac{15}{2} h_-^{-4} (x)^4 + 4h_-^{-3} (x)^3$$

$$\left(\frac{d}{dx} u_1^{0+}, \frac{d}{dx} u_{1+1}^{1-}\right)_{x_1}^{x_2} = \frac{18}{5} h_+^{-5} (x)^5 - \frac{15}{2} h_+^{-4} (x)^4 + 4h_+^{-3} (x)^3$$

$$\left(\frac{d}{dx} u_1^{1+}, \frac{d}{dx} u_1^{0+}\right)_{x_1}^{x_2} = -\frac{18}{5} h_+^{-5} (h_+ - x)^5 + \frac{15}{2} h_+^{-4} (h_+ - x)^4 - 4h_+^{-3} (h_+ - x)^3$$

$$\left(\frac{d}{dx} u_1^{0+}, \frac{d}{dx} u_1^{1+}\right)_{x_1}^{x_2} = \left(\frac{d}{dx} u_1^{1+}, \frac{d}{dx} u_1^{0+}\right)_{x_1}^{x_2}$$

$$\left(\frac{d}{dx} u_1^{1-}, \frac{d}{dx} u_1^{0-}\right)_{x_1}^{x_2} = -\frac{18}{5} h_-^{-5} (x)^5 + \frac{15}{2} h_-^{-4} (x)^4 - 4h_-^{-3} (x)^3$$

$$\left(\frac{d}{dx} u_1^{0-}, \frac{d}{dx} u_1^{1-}\right)_{x_1}^{x_2} = \left(\frac{d}{dx} u_1^{1-}, \frac{d}{dx} u_1^{0-}\right)_{x_1}^{x_2}$$

$$\left(\frac{d}{dx} u_1^{1+}, \frac{d}{dx} u_{1+1}^{0-}\right)_{x_1}^{x_2} = -\frac{18}{5} h_+^{-5} (x)^5 + \frac{21}{2} h_+^{-4} (x)^4 - 10 h_+^{-3} (x)^3 + 3h_+^{-2} (x)^2$$

$$\left(\frac{d}{dx} u_1^{0-}, \frac{d}{dx} u_{1-1}^{1+}\right)_{x_1}^{x_2} = -\frac{18}{5} h_-^{-5} (x)^5 + \frac{21}{5} h_-^{-4} (x)^4 - 10 h_-^{-3} (x)^3 + 3h_-^{-2} (x)^2$$

$$\left(\frac{d}{dx} u_1^{1-}, \frac{d}{dx} u_{1-1}^{1+}\right)_{x_1}^{x_2} = \frac{9}{5} h_-^{-4} (x)^5 - \frac{9}{2} h_-^{-3} (x)^4 + \frac{11}{3} h_-^{-2} (x)^3 - h_-^{-1} (x)^2$$

$$\left(\frac{d}{dx} u_i^{1+}, \frac{d}{dx} u_{i+1}^{1-}\right)_{x_1}^{x_2} = \frac{9}{5} h_+^{-4} (x)^5 - \frac{9}{2} h_+^{-3} (x)^4 + \frac{11}{3} h_+^{-2} (x)^3 - h_+^{-1} (x)^2$$

$$\left(\frac{d}{dx} u_i^{1-}, \frac{d}{dx} u_i^{1-}\right)_{x_1}^{x_2} = \frac{9}{5} h_+^{-4} (x)^5 - 3 h_-^{-3} (x)^4 + \frac{4}{3} h_-^{-2} (x)^3$$

$$\left(\frac{d}{dx} u_i^{1+}, \frac{d}{dx} u_i^{1+}\right)_{x_1}^{x_2} = -\frac{9}{5} h_+^{-4} (h_+ - x)^5 + 3 h_+^{-3} (h_+ - x)^4 - \frac{4}{3} h_+^{-2} (h_+ - x)^3$$

APPENDIX C

THE CODE CHDC.1) General description

CHD (Cubic Hermite Diffusion) solves the multigroup diffusion equations in a two-dimensional, rectangular geometry by expanding piecewisely the unknown solutions in bicubic Hermite polynomials and using the Galerkin method to arrive at a discrete formulation. The resulting equations are uncoupled through the power iteration method, and the in-group solutions are achieved by direct inversion (Cholesky factorization).

When the solution is converged several output options are available: fine flux expansion inside the coarse mesh, cross sections averaging, reaction rates and average fluxes calculations. All these outputs are normalized to an specified overall fission rate.

In Fig. C.1 we show the general code logic.

The variable dimensioning techniques available in Fortran IV are used in CHD. Practically all dimensioned variables are stored in a general, one-dimensional array called A and dimensioned in a small main program.

A rectangular coarse mesh partition of the problem domain is assumed. At each point CHD will construct the appropriate set of bicubic basis functions depending on several conditions such as boundary conditions, user specification, or neighboring regions characteristics. Current continuity is imposed where appropriate.

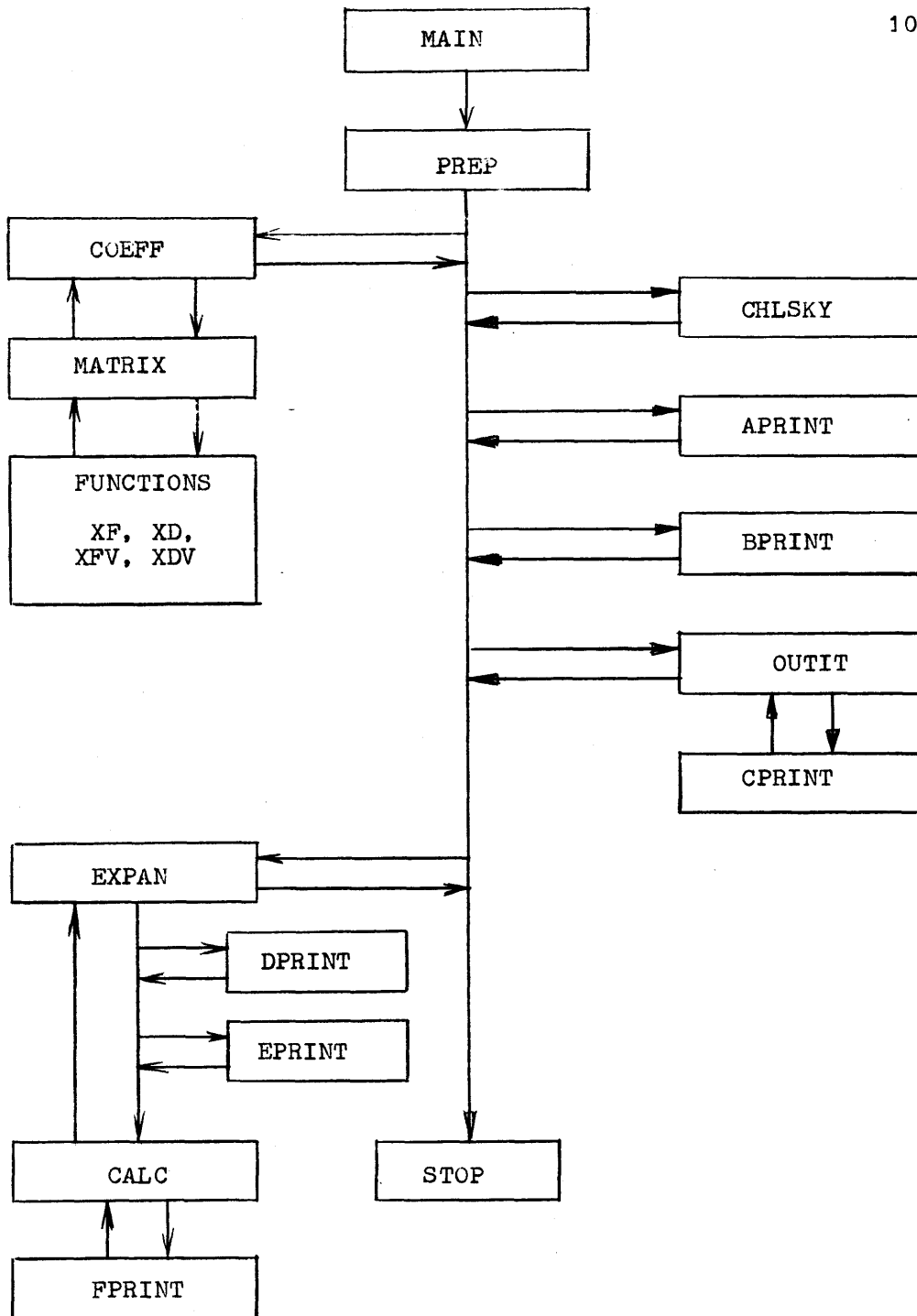


Fig. C.1 - CHD basic logic

The coordinate system orientation shown in Fig. C.2 is assumed by CHD.

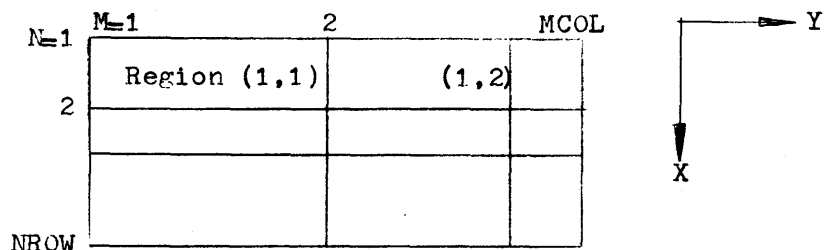


Fig. C.2 - Coordinate system orientation assumed by CHD.

The minimum number of rows or columns allowed is 2, and a column or a row has to be specified at each boundary. Only convex regions are allowed so that the problem domain is always a full rectangle. Zero flux and zero current boundaries are allowed.

The ordering of the unknowns is assumed to be done in the following way:

- 1) All rows
- 2) All columns
- 3) All unknowns at each point

Unknowns at each point are assumed to be ordered as shown below:

- 1.1) Flux
- 1.2) Current in X direction or

1.2.1)	Current in negative X direction.
1.2.2)	Current in positive X direction.

- 1.3) Current in Y direction or
- 1.3.1) Current in negative Y direction.
 - 1.3.2) Current in positive Y direction.

1.4) Mixed derivative.

At each corner point one will have 1 unknown, at each non-corner boundary point 2 or 3, and in each internal point 4, 5 or 6.

Any number of energy groups greater than, or equal to, 2 may be specified. As restrictions, however, all fissions will lead to neutrons being born in the highest energy group and any outscattering will lead to the group immediately below. Groups are numbered beginning with the highest energy one.

Next we present a summary of the main features of the subroutines.

MAIN: The size of the one-dimensional array A has to be specified according to the following minimum value (see definition of variables in input description).

$$\begin{aligned}
 M(\min) = & LMAX (6 KGROUP + 1.5) + 1.5 LXMAX \\
 & + LXCMAX (2 KGROUP + 0.5) \\
 & + LXFMAX (2 KGROUP - 1.5) + JMAX (4 KGROUP -1) \\
 & + JVMAX (NVN * MVM/2 + NVN + MVM + 1) \\
 & + NROW * MCOL (KGROUP + 7) \\
 & + (NROW + MCOL) /2 + 7 KGROUP + 4 \qquad (C.1)
 \end{aligned}$$

Neglect any fraction.

The statement

$$MSIZE = M$$

should also be set to this value.

We next show some values for variables that were arrived at for problems shown in the thesis in order to provide for some insight.

Table C.1 Some Variable Values

<u>Problems</u>	<u>LMAX</u>	<u>LXMAX</u>	<u>LXCMAX</u>	<u>LXFMAX</u>	<u>M Size (mim.)</u>
Ref. A. sec. 3.4	104	1350	2291	2596	20632
Case 1, sec. 3.4	18	123	145	228	1955
Ref, sec. 3.5	146	2019	3770	3892	33912
Case 3, sec.3.1	72	844	1344	1616	
Zion 1 (6x6)	128	1558	3183	2988	26496
(7x7)	190	2443	5775	4696	44626
(10x10)	348	5223	13814	10098	100945

The CHD object deck requires approximately 106 k bytes, and about all arrays are stored in MSIZE. Thus, the amount of fast core required is, approximately

$$\text{Fast core (bytes)} = \text{MSIZE} * 4 + 106000$$

PREP: Is the preparation and program control subroutine.

- a) Reads almost all data
- b) Calculates number of bicubic functions at each point, normalization factor θ where necessary, imposes boundary conditions at the expansion functions where appropriate, and orders the unknowns.

COEFF: Sweeps through all the unknowns and at each one sets up all the unknowns which will couple with the first one. Then calls MATRIX and sets up one-dimensional (in each group) arrays in which the non-zero elements of all coefficient matrices will be stored. If the matrix is symmetric only the lower diagonal part is stored.

MATRIX: Performs the actual calculations of the coefficients with use of the following functions:

XF - bicubic basis functions inner products over the whole range of the element.

XD - derivative of basis functions inner products over the whole range of the element.

These inner products are listed in Ref. 7 App. B.

XFV - bicubic basis functions inner products over partial intervals of the element.

XDV - derivative of bicubic basis functions inner products over partial intervals of the element.

The formulæ for these last inner products are given in App. B of this thesis.

CHLSKY: Calculates the lower triangular matrices corresponding to the Cholesky factorization.

OUTIT: Reads initial flux guess (if any), iterates according to the power iteration method and inverts the in-group matrices using the factorized Cholesky matrices.

EXPAN: Normalizes the solution to an specified fission rate, calculates region-wise reaction rates and average fluxes. These calculations can only be done over each element, there being no provision for specifying an output region over more than one mesh.

Reaction rates are defined as

$$\text{ABSRP} = \sum_{K=1}^{KG-1} \int_V (\Sigma_r(K, \underline{r}) - \Sigma_s(K, \underline{r})\phi(K, \underline{r})) d\underline{r} \\ + \int_V \Sigma_r(KG, \underline{r})\phi(KG, \underline{r}) d\underline{r}$$

$$\text{TRNSP} = \sum_{K=1}^{KG} \int_V (1/D(K, \underline{r})\phi(K, \underline{r})) d\underline{r}$$

Where $KG \equiv \text{KGROUP}$

A collapsed macroscopic cross section is defined as

$$\bar{\Sigma}(K, V) = \frac{\int_V \Sigma(K, \underline{r})\phi(K, \underline{r}) d\underline{r}}{\bar{\phi}(K, V)V}$$

with the average flux defined as

$$\bar{\phi}(K, V) = \frac{\int_V \phi(K, \underline{r}) d\underline{r}}{V}$$

The collapsed diffusion coefficient is defined as

$$[\bar{D}(K, V)]^{-1} = \frac{\int_V (1/D(K, \underline{r})\phi(K, \underline{r})) d\underline{r}}{\bar{\phi}(K, V)V}$$

CALC: Calculates the fine flux distribution at equally spaced points inside the coarse mesh.

All XPRINT sub-routines do just printing, except EPRINT where the region-wise collapsed cross sections are calculated.

C.2) Input Preparation

Statements 1 and 2 in MAIN

Set dimension of A and value of MSIZE to M or more, according to Eq. (C.1).

Card 1 - Format I5

NRUN - Number of cases to be performed. Every case will require the input of all the data, sequentially.

Card 2 - 14I5

NROW - number of rows (including boundary)

MCOL - number of columns (including boundary)

KGROUP - number of groups

JMAX - number of different material compositions to be read in.

(*,**) LMAX - number of unknowns per group

(*,**) LXMAX - size of the one-dimensional array where the lower triangular part of matrix F1 is stored (see text).

(*,**) LMCMAX - size of the one-dimensional array (in each group) where the lower triangular part of the Cholesky matrices will be stored (see text).

(*,**) LMFMAX - size of the one-dimensional arrays (in each group) where the F(2)...F(KGROUP) and S(1)...S(KGROUP-1) matrices will be stored (see text).

JVMAX - number of regions with varying parameters to be specified.

(*) NVN - maximum number of internal rows specified for any of the JVMAX regions.

(*) MVM - maximum number of internal columns specified for any of the JVMAX regions.

(*) Values used just for dimensioning purposes at this stage.
(**) Actual values are printed.

CHD will not begin any operation and will print an error message if the required value for M is less than specified in MSIZE.

If no variable parameter region is to be specified, JVMAX, NVN and MVM should be input as zero.

Card 3 - 1615

NCONV - maximum number of outer iterations allowed.

If this number is reached without convergence, the calculation is terminated with all output.

ICMAX - (=0) - no fine flux is to be calculated inside the coarse meshes.

(<0) - fine flux is calculated at all meshes at (-ICMAX) equally spaced internal points.

(>0) - user will specify ICMAX sets where he wants the fine flux to be calculated. In this case, cards 22 have to be specified, and flux at remaining meshes will be calculated at 5 equally spaced internal points.

NP - (≤ 0) - no initial guess of the solution is to be input. Code will start with flat flux at all points except at a boundary with a zero flux boundary condition.

(≥ 0) - user will provide an initial solution guess. Cards 21 have to be input.

IOUT - (< 0) - user does not want the solution punched out.

(≥ 0) - user wants solution to be punched out in Format 6E12.5.

ICLPS - (< 0) - code will collapse all cross sections at all regions with varying parameters.

(≥ 0) - user will specify ICLPS regions where he wants the cross sections to be collapsed. Cards 23 have to be input.

The collapsing has only sense over regions with varying parameters, obviously.

Card 4 - 8E10.8

BL - left boundary condition (1.0 - zero derivative; 0.0 zero flux).

BR - right boundary condition.

BT - top boundary condition

BB - bottom boundary condition.

Card 5-8E10.5

EPSCH - Cholesky rejection criterion. Code will reject any value less than EPSCH in the calculation of the Cholesky matrices.

EPSF - Solution convergence criterion

EPSL - Eigenvalue convergence criterion.

CHD will assume the problem as converged when

$$\frac{\left| \left\| \underline{a}^{(t-1)} \right\|_{\infty} - \left\| \underline{a}^{(t)} \right\|_{\infty} \right|}{\left\| \underline{a}^{(t-1)} \right\|_{\infty}} < \text{EPSF}$$

and

$$\frac{\left| \lambda^{(t-1)} - \lambda^{(t)} \right|}{\lambda^{(t-1)}} < \text{EPSL}$$

Cards 6 - 3I5

Through cards 6 the user is able to specify the number and type of the expansion functions he wants at each mesh point, with the few exceptions listed below. In each card one point is specified as:

N - row number

M - column number

IPP - any allowed expansion function option (40,50, 60,41,51,61,421 and 422).

CHD reads type-6 cards until a blank field I5 is found.

If the user wants the code to use its default options (probably a good option when first running a problem) he should input just one blank card.

Expansion Functions Option Definition

The first digit of the option will specify the number of expansion functions at the particular point. The third and/or second digits will specify the type of continuity assumed.

In Table C.1 we show all the allowed options.

Table C.2 - Expansion Functions Option Definition

SET	Continuity Condition for			
	Flux	First derivative in X direction	First derivative in Y direction	Mixed derivative
40	Yes	$\partial / \partial x$	$\partial / \partial y$	$\partial^2 / \partial x \partial y$
41	Yes	$\partial / \partial x$	$\partial / \partial y$	$(\theta/D)(\partial^2 / \partial x \partial y)$
421	Yes	$(\theta/D)(\partial / \partial x)$	$\partial / \partial y$	$(\theta/D)(\partial^2 / \partial x \partial y)$
422	Yes	$\partial / \partial x$	$(\theta/D)(\partial / \partial y)$	$(\theta/D)(\partial^2 / \partial x \partial y)$
50	Yes	Open	∂ / ∂	$\partial^2 / \partial x \partial y$
		$\partial / \partial x$	Open	
51	Yes	Open	$\partial / \partial y$	$(\theta/D)(\partial^2 / \partial x \partial y)$
		$\partial / \partial x$	Open	
60	Yes	Open	Open	$\partial^2 / \partial x \partial y$
61	Yes	Open	Open	$(\theta/D)(\partial^2 / \partial x \partial y)$

In options 50 and 51 the code will decide in which direction the opening should be performed.

Input options which are not allowed

a) Corner points should not be specified because there is no reason for doing so. The code will always assume a 40 set, afterwards neglecting 3 of the 4 functions depending on the boundary conditions.

b) At the remaining boundary points no 60 or 61 sets should be specified because they are equivalent to 50 and 51 sets, respectively, in these cases. The latter should be used.

c) When inputting 421 or 422 options at boundary points the user should be careful in specifying the direction of the bending (defined by the last digit, 1 - x direction, 2 - y direction). If a mistake is made the code will use its default option.

Defaults

The code will use the diffusion coefficients in the immediate surroundings of a point to evaluate its characteristics. In Fig. C.3 we show a general arrangement around a point.

Table C.3 Default sets for expansion functions

Condition		Default set
$D_1 = D_2 = D_3 = D_4$		40
$D_1 = D_4$ $D_2 = D_3$ $D_1 \neq D_2$	$D_I(\underline{r}) = D_{IV}(\underline{r}) = \text{const.}$ and $D_{II}(\underline{r}) = D_{III}(\underline{r}) = \text{const.}$	422
	Otherwise	51 (with uncoupling in X direction).
$D_1 = D_2$ $D_3 = D_4$ $D_1 \neq D_3$	$D_I(\underline{r}) = D_{II}(\underline{r}) = \text{const.}$ and $D_{III}(\underline{r}) = D_{IV}(\underline{r}) = \text{const.}$	421
	Otherwise	51 (with uncoupling in Y direction)
$D_1 D_2 \neq D_3 D_4$	$D_I(\underline{r})$ and $D_{II}(\underline{r})$ and $D_{III}(\underline{r})$ and $D_{IV}(\underline{r})$ are constant	41
	Otherwise	61

Card 7 - 8E10.8

XNORM - Overall fission rate to which all output
will be normalized.

Card 8 - 8E10.8

XI(K), K = 1, KGROUP

- Neutron emission per fission in group K.

These values will be used only to calculate the fission rate
in the normalization process.

Card 9 - I5

J - Composition number.

Card 10 - 8E10.8

DI(K,J), K = 1, KGROUP

- diffusion coefficient (cm) for group K and composition
J.

Card 11 - 8E10.8

SR(K,J), K = 1, KGROUP

- removal cross section (/cm)

Card 12 - 8E10.8

SF(K,J), K = 1, KGROUP

- fission production cross section $\nu\Sigma_f$ (/cm)

Card 13 - 8E10.8

ST(K,J), K = 1, GROUP

- Outscattering cross section (/cm) from group K to
K+1.

JMAX sets of cards 9 through 13 are expected.

Card 20

JCV (J,NV,MV), MV = 1, (MVMAX (J) + 1)

Composition number of sub-region (NV,MV) of region J.

(NVMAX(J) + 1) Cards 20 are expected.

JVMAX sets of cards 17 through 20 are expected.

Cards 21 - 6E12.5 (only present if NP>0)

Y(K,L), L = 1, LMAX

KGROUP sets of cards 21 are expected.

Cards 22 - 14I5 (only present if ICMAX>0)

ISTR (JJ) - number of internal equally spaced sub-rows
to be considered in region JJ.

ISTC (JJ) - number of internal equally spaced sub-columns
to be considered in region JJ.

MINR (JJ) - row identification of region JJ.

MINC (JJ) - column identification of region JJ.

KC (JJ) - energy group.

ICMAX cards 22 are expected.

Cards 23 - 14I5 (only present if ICLPS>0)

N - row identification of region where cross sections
collapsing is to be done.

M - column identification of region where cross sections
collapsing is to be done.

Card 14 - 8E10.8

HN(N), N = 1, (NROW-1)

- mesh spacing in X direction.

Card 15 - 8E10.8

HM(M), M = 1, (MCOL-1)

- mesh spacing in Y direction.

Card 16 - 16I5

JC(N,M), M = 1, (MCOL-1)

- composition characteristics of region N,M.

(<0) - code assumes a region with constant properties
and corresponding to composition (-JC)(>0) - code assumes a region with varying properties; the
region JC will be specified in cards 17 through 20.

(NROW-1) cards 16 are expected.

Cards 17 through 20 are not required if JVMAX = 0 and NVN = 0
and MVM = 0.Card 17 - 3I5

J - region with varying parameters.

NVMAX(J) - number of sub-rows in region J.

MVMAX(J) - number of sub-columns in region J.

Card 18 - 8E10.8

HNV (J,NV), NV = 1, (NVMAX(J) + 1)

- Row sub-mesh.

Card 19 - 8E10.8

HNV (J,MV), MV = 1, (MVMAX(J) + 1)

- Column sub-mesh.

APPENDIX D

SOURCE LISTING OF CHD, SAMPLE INPUT
AND OUTPUT (only in MIT library copies)

1										
3	3	2	5	150	2100	4000	4100	1	3	3
50	-4	0	0	-1						
	1.0		0.0		1.0		0.0			
	1.0E-06		1.0E-04		1.0E-04					
	100.0									
	1.0		1.0							
1										
	1.2		0.15							
	0.101		0.02							
	0.1									
2										
	1.5		0.4							
	0.0623		0.2							
	0.0		0.218							
	0.06									
3										
	1.55		0.4133							
	0.0623		0.25							
	0.0		0.218							
	0.06									
4										
	1.60		0.4267							
	0.0623		0.3							
	0.0		0.218							
	0.06									
5										
	1.65		0.44							
	0.0623		0.35							
	0.0		0.218							
	0.06									
	20.0		20.0							
	20.0		20.0							
1	-1									
-1	-1									
1	3	3								
	5.0		5.0		5.0		5.0			
	5.0		5.0		5.0		5.0			
5	4	3	2							
4	4	3	2							
3	3	3	2							
2	2	2	2							

```

$JOB          DEPPE, TIME=2
1  DIMENSION A(40000)
2  MSIZE=40000
3  READ(5,1000)NRUN
4  NRUN1=0
5  990 NRUN1=NRUN1+1
6  READ(5,1000) NROW,MCOL,KGROUP,JMAX,LMAX,LXMAX,LXCMAX,LXFMAX,JVMAX,
1 NVN,MVM
7  1000 FORMAT(14I5)
8  NROW1=NROW-1
9  MCOL1=MCOL-1
10 K1=KGROUP-1
11 K2=3*KGROUP*LMAX
12 NVN=NVN+1
13 MVM=MVM+1
14 IF(JVMAX.EQ.0) JVMAX=1
15 L1=1
16 L2=L1+2*KGROUP*LXCMAX
17 L3=L2+6*KGROUP*LMAX
18 L4=L3+LXMAX
19 L5=L4+LXFMAX*K1
20 L6=L5+LXFMAX*K1
21 L7=L6+KGROUP
22 L8=L7+JMAX*KGROUP
23 L9=L8+JMAX*KGROUP
24 L10=L9+JMAX*KGROUP
25 L11=L10+JMAX*K1
26 L12=L11+NROW1
27 L12A=L12+MCOL1
28 L12B=L12A+JVMAX*NVN
29 L12C=L12B+JVMAX*MVM
30 L12D=L12C+KGROUP
31 L12E=L12D+KGROUP
32 L12F=L12E+KGROUP
33 L12G=L12F+KGROUP
34 L12H=L12G+KGROUP
35 L13=L12H+KGROUP
36 L14=L13+NROW*MCOL*6
37 L15=L14+NROW*MCOL*KGROUP
38 L16=L15+(LMAX+1)/2
39 L17=L16+(LMAX+1)/2
40 L18=L17+(LMAX+1)/2
41 L19=L18+(LXMAX+1)/2
42 L20=L19+(LXCMAX+1)/2
43 L21=L20+(LXFMAX+1)/2
44 L22=L21+(NROW1*MCOL1+1)/2
45 L23=L22+(JVMAX*NVN*MVM+1)/2
46 L24=L23+(JVMAX+1)/2
47 L25=L24+(JVMAX+1)/2
48 MSIZE1=L25+(NROW*MCOL+1)/2
49 IF(MSIZE.LT.MSIZE1) GOTO 1100
50 CALL PREP(NRUN1,K2,
1 MSIZE,MSIZE1, NROW,NROW1,MCOL,MCOL1,KGROUP,K1,JMAX,
1 JVMAX,NVN,MVM,LMAX,LXMAX,
1 LXCMAX,LXFMAX,A(L1),A(L2),A(L3),A(L4),A(L5),A(L6),A(L7),A(L8),
1 A(L9),A(L10),A(L11),A(L12),A(L12A),A(L12B),A(L12C),A(L12D),

```

```
2A(L12E),A(L12F),A(L12G),A(L12H),
2      A(L13),A(L14),A(L15),A(L16),A(L17),A(L18),
3A(L19),A(L20),A(L21),A(L22),A(L23),A(L24),A(L25))
51  IF(NRUN1.LT.NRUN) GOTO 990
52  GOTO 1200
53  1100 M=MSIZE1-MSIZE
54  WRITE(6,1150) M
55  1150 FORMAT(1H1,'***ERROR*** SIZE OF ARRAY A IS TOO SMALL BY',I 6,3X,
56  1200 STOP
57  END
```

```

58     SUBROUTINE PREP(NRUN1,K2,MSIZE,MSIZE1,NROW,NROW1,MCOL,MCOL1,
1KGROUP,K1,JMAX,JVMAX,NVN,MVM,LMAX,
1LLXMAX,LXCMAX,LXFMAX,A,AR,F1,F,S,XI,DI,SR,SF,ST,HN,HM,HNV,HMV,
2DA,DAA,DB,DBB,D1,D2,IND,TH,LXD,
2LXDC,LXDF,LLX,LLXC,LLXF,JC,JCV,NVMAX,MVMAX,IP)
59     REAL*8 A(KGROUP,LXCMAX),AR(K2)
60     INTEGER*2 LXDC(LMAX),LXDF(LMAX),LLX(LXMAX),LLXC(LXCMAX),
1LLXF(LXFMAX),JC(NROW1,MCOL1)
61     INTEGER*2 JCV(JVMAX,NVN,MVM),NVMAX(JVMAX),MVMAX(JVMAX)
62     INTEGER*2 IP(NROW,MCOL)
63     DIMENSION XI(KGROUP),DI(KGROUP,JMAX),SR(KGROUP,JMAX),
1SF(KGROUP,JMAX),ST(K1,JMAX)
64     DIMENSION HN(NROW1),HM(MCOL1)
65     DIMENSION IND(NROW,MCOL,6),TH(KGROUP,NROW,MCOL)
66     DIMENSION F1(LXMAX),F(K1,LXFMAX),S(K1,LXFMAX)
67     DIMENSION HNV(JVMAX,NVN),HMV(JVMAX,MVM),DA(KGROUP),DAA(KGROUP),
1DB(KGROUP),DBB(KGROUP),D1(KGROUP),D2(KGROUP)
68     READ(5,1000) NCONV,      ICMAX, NP, IOUT, ICLPS
69     1000 FORMAT(16I5)
70     READ(5,1010) BL, BR, BT, BB
71     READ(5,1010) EPSCH, EPSF, EPSL
72     1010 FORMAT(8E10.8)
73     DO 1012 N=1,NROW
74     DO 1012 M=1,MCOL
75     IP(N,M)=42
76     IND(N,M,1)=110
77     IND(N,M,2)=210
78     IND(N,M,3)=0
79     IND(N,M,4)=120
80     IND(N,M,5)=0
81     1012 IND(N,M,6)=220
82     IP(1,1)=40
83     IP(1,MCOL)=40
84     IP(NROW,1)=40
85     IP(NROW,MCOL)=40
86     ISMAX=NROW*MCOL+10
87     DO 1024 IS=1,ISMAX
88     READ(5,1000) N,M,IPP
89     IF(N.EQ.0) GOTO 1025
90     IP(N,M)=IPP
91     II=IP(N,M)/10
92     IF(II.EQ.4) GOTO 1024
93     IND(N,M,2)=211
94     IND(N,M,3)=212
95     IND(N,M,4)=121
96     IND(N,M,5)=122
97     1024 CONTINUE
98     1025 CONTINUE
99     READ(5,1010) XNORM
100    READ(5,1010) (XI(K),K=1,KGROUP)
101    DO 1050 JJ=1,JMAX
102    READ(5,1000) J
103    READ(5,1010) (DI(K,J),K=1,KGROUP)
104    READ(5,1010) (SR(K,J),K=1,KGROUP)
105    READ(5,1010) (SF(K,J),K=1,KGROUP)
106    READ(5,1010) (ST(K,J),K=1,K1)

```

```

107 1050 CONTINUE
108 READ(5,1010)(HN(N),N=1,NROW1)
109 READ(5,1010)(HM(M),M=1,MCOL1)
110 DO 1060 N=1,NROW1
111 1060 READ(5,1000)(JC(N,M),M=1,MCOL1)
112 IF(JVMAX.EQ.1.AND.NVN.EQ.1.AND.MVM.EQ.1) GOTO 1083
113 DO 1082 JJ=1,JVMAX
114 READ(5,1000) J,NVMAX(J),MVMAX(J)
115 NVMAX1=NVMAX(J)+1
116 MVMAX1=MVMAX(J)+1
117 READ(5,1010)(HNV(J,NV),NV=1,NVMAX1)
118 READ(5,1010)(HNV(J,MV),MV=1,MVMAX1)
119 DO 1082 NV=1,NVMAX1
120 READ(5,1000)(JCV(J,NV,MV),MV=1,MVMAX1)
121 1082 CCONTINUE
122 1083 CONTINUE
123 DO 1421 K=1,KGROUP
124 DO 1421 N=1,NROW
125 DO 1421 M=1,MCOL
126 1421 TH(K,N,M)=-1000.
127 IF(NROW1.LT.2.OR.MCOL1.LT.2) GOTO 1750
128 DO 1700 N=2,NROW1
129 DO 1700 M=2,MCOL1
130 IF(IP(N,M).EQ.40.CR.IP(N,M).EQ.60) GOTO 1700
131 JA=JC(N,M-1)
132 JAA=JC(N-1,M-1)
133 JB=JC(N,M)
134 JBB=JC(N-1,M)
135 IF(JA ) 1430,1430,1450
136 1430 DO 1440 K=1,KGROUP
137 1440 DA(K)=DI(K,-JA)
138 GOTO 1470
139 1450 MVA=MVMAX(JA )+1
140 JA=JCV(JA,1,MVA)
141 DO 1460 K=1,KGROUP
142 1460 DA(K)=DI (K,JA)
143 1470 IF(JAA ) 1480,1480,1500
144 1480 DO 1490 K=1,KGROUP
145 1490 DAA(K)=DI(K,-JAA)
146 GOTO 1520
147 1500 NVA=NVMAX(JAA )+1
148 MVA=MVMAX(JAA )+1
149 JAA=JCV(JAA,NVA,MVA)
150 DO 1510 K=1,KGROUP
151 1510 DAA(K)=DI (K,JAA)
152 1520 IF(JB ) 1530,1530,1550
153 1530 DO 1540 K=1,KGROUP
154 1540 DB(K)=DI(K,-JB)
155 GOTO 1570
156 1550 JB=JCV(JB,1,1)
157 DO 1560 K=1,KGROUP
158 1560 DB(K)=DI (K,JB)
159 1570 IF(JBB ) 1580,1580,1600
160 1580 DO 1590 K=1,KGROUP
161 1590 DBB(K)=DI(K,-JBB)
162 GOTO 1620

```

```

163 1600 NVA=NVMAX(JBB )+1
164      JBB=JCV(JBB,NVA,1)
165      DO 1610 K=1,KGROUP
166 1610 DBB(K)=DI (K,JBB)
167 1620 IF(IP(N,M).EQ.50.CR.IP(N,M).EQ.51.OR.IP(N,M).EQ.61) GOTO 1670
168      IF(IP(N,M).EQ.41) GOTO 1675
169      IF(IP(N,M).EQ.421) GOTO 1655
170      IF(IP(N,M).EQ.422) GOTO 1635
171      ABS1=DA(KGROUP)/DB(KGROUP)
172      ABS2=DAA(KGROUP)/DBB(KGROUP)
173      IF(ABS(ABS1-ABS2).GT.1.0E-04) GOTO 1663
174      IF(ABS(1.0-ABS1).GT.1.0E-04) GOTO 1630
175      ABS1=DA(KGROUP)/DAA(KGROUP)
176      IF(ABS(1.0-ABS1).GT.1.0E-04) GOTO 1650
177      IP(N,M)=40
178      GOTO 1700
179 1630 IP(N,M)=422
180      IF(JA.LT.0.AND.JB.LT.0) GOTO 1635
181      IP(N,M)=51
182      IND(N,M,4)=121
183      IND(N,M,5)=122
184 1635 DO 1640 K=1,KGROUP
185 1640 TH(K,N,M)=(DA(K)+CB(K))/2.0
186      GOTO 1700
187 1650 IP(N,M)=421
188      IF(JA.LT.0.AND.JAA.LT.0) GOTO 1655
189      IP(N,M)=51
190      IND(N,M,2)=211
191      IND(N,M,3)=212
192 1655 DO 1660 K=1,KGROUP
193 1660 TH(K,N,M)=(DA(K)+CAA(K))/2.0
194      GOTO 1700
195 1663 IP(N,M)=41
196      IF(JA.LT.0.AND.JAA.LT.0.AND.JB.LT.0.AND.JBB.LT.0) GOTO 1675
197      IP(N,M)=61
198      IND(N,M,2)=211
199      IND(N,M,3)=212
200      IND(N,M,4)=121
201      IND(N,M,5)=122
202      GOTO 1675
203 1670 IF(IP(N,M).EQ.61) GOTO 1675
204      IF(ABS(DA(KGROUP)-DAA(KGROUP)).LT.1.0E-04) GOTO 1672
205      IND(N,M,4)=120
206      IND(N,M,5)=0
207      GOTO 1673
208 1672 IND(N,M,2)=210
209      IND(N,M,3)=0
210 1673 IF(IP(N,M).EQ.50) GOTO 1700
211 1675 DO 1680 K=1,KGROUP
212 1680 TH(K,N,M)=(DA(K)+DAA(K)+DB(K)+DBB(K))/4.0
213 1700 CONTINUE
214 1750 CCNTINUE
215      IF(MCOL1.LT.2) GOTO 2150
216      DO 2100 M=2,MCOL1
217      IF(IP(1,M).EQ.40) GOTO 1930
218      IF(IP(1,M).EQ.50) GOTO 1925

```

```

219      JA=JC(1,M-1)
220      JB=JC(1,M)
221      IF(JA      ) 1810,1810,1830
222 1810 DO 1820 K=1,KGROUP
223 1820 DA(K)=DI(K,-JA)
224      GOTO 1850
225 1830 MVA=MVMAX(JA      )+1
226      JA=JCV(JA,1,MVA)
227      DO 1840 K=1,KGROUP
228 1840 DA(K)=DI (K,JA)
229 1850 IF(JB      ) 1860,1860,1880
230 1860 DO 1870 K=1,KGROUP
231 1870 DB(K)=DI(K,-JB)
232      GOTO 1900
233 1880 JB=JCV(JB,1,1)
234      DO 1890 K=1,KGROUP
235 1890 DB(K)=DI (K,JB)
236 1900 IF(IP(1,M).EQ.41.OR.IP(1,M).EQ.51) GOTO 1915
237      IF(IP(1,M).EQ.422) GOTO 1915
238      IF(ABS(DA(KGROUP)-DB(KGROUP)).GT.1.0E-04.AND.BT.GT.0.5)GOTO 1910
239      IP(1,M)=40
240      GOTO 1930
241 1910 IP(1,M)=422
242      IF(JA.LT.0.AND.JB.LT.0) GOTO 1915
243      IP(1,M)=51
244      IND(1,M,4)=121
245      IND(1,M,5)=122
246 1915 DO 1920 K=1,KGROUP
247 1920 TH(K,1,M)=(DA(K)+DB(K))/2.0
248 1925 IND(1,M,2)=210
249      IND(1,M,3)=0
250 1930 CONTINUE
251      IF(IP(NROW,M).EQ.40) GOTO 2100
252      IF(IP(NROW,M).EQ.50) GOTO 2095
253      JA=JC(NROW1,M-1)
254      JB=JC(NROW1,M)
255      IF(JA      ) 1950,1950,1970
256 1950 DO 1960 K=1,KGROUP
257 1960 DA(K)=DI(K,-JA)
258      GOTO 1990
259 1970 NVA=NVMAX(JA      )+1
260      MVA=MVMAX(JA      )+1
261      JA=JCV(JA,NVA,MVA)
262      DO 1980 K=1,KGROUP
263 1980 DA(K)=DI (K,JA)
264 1990 IF(JB      ) 2000,2000,2020
265 2000 DO 2010 K=1,KGROUP
266 2010 DB(K)=DI(K,-JB)
267      GOTO 2040
268 2020 NVA=NVMAX(JB      )+1
269      JB=JCV(JB,NVA,1)
270      DO 2030 K=1,KGROUP
271 2030 DB(K)=DI (K,JB)
272 2040 IF(IP(NROW,M).EQ.41.OR.IP(NROW,M).EQ.51) GOTO 2055
273      IF(IP(NROW,M).EQ.422) GOTO 2055
274      IF(ABS(DA(KGROUP)-DB(KGROUP)).GT.1.0E-04.AND.BB.GT.0.5) GOTO 2050

```

```

275         IP(NROW,M)=40
276         GOTO 2100
277 2050    IP(NROW,M)=422
278         IF(JA.LT.0.AND.JB.LT.0) GOTO 2055
279         IP(NROW,M)=51
280         IND(NROW,M,4)=121
281         IND(NROW,M,5)=122
282 2055    DO 2060 K=1,KGROUP
283 2060    TH(K,NROW,M)=(DA(K)+DB(K))/2.0
284 2095    IND(NROW,M,2)=210
285         INC(NROW,M,3)=0
286 2100    CONTINUE
287 2150    CONTINUE
288         IF(NROW1.LT.2) GOTO 2500
289         DO 2480 N=2,NROW1
290         IF(IP(N,1).EQ.40) GOTO 2350
291         IF(IP(N,1).EQ.50) GOTO 2345
292         JA=JC(N-1,1)
293         JB=JC(N,1)
294         IF(JA ) 2210,2210,2230
295 2210    DO 2220 K=1,KGROUP
296 2220    DA(K)=DI(K,-JA)
297         GOTO 2250
298 2230    NVA=NVMAX(JA )+1
299         JA=JCV(JA,NVA,1)
300         DO 2240 K=1,KGROUP
301 2240    DA(K)=DI (K,JA)
302 2250    IF(JB ) 2260,2260,2280
303 2260    DO 2270 K=1,KGROUP
304 2270    DB(K)=DI(K,-JB)
305         GOTO 2300
306 2280    JB=JCV(JB,1,1)
307         DO 2290 K=1,KGROUP
308 2290    DB(K)=DI (K,JB)
309 2300    IF(IP(N,1).EQ.41.OR.IP(N,1).EQ.51) GOTO 2315
310         IF(IP(N,1).EQ.421) GOTO 2315
311         IF(ABS(DA(KGROUP)-DB(KGROUP)).GT.1.0E-04.AND.BL.GT.0.5) GOTO 2310
312         IP(N,1)=40
313         GOTO 2350
314 2310    IP(N,1)=421
315         IF(JA.LT.0.AND.JB.LT.0) GOTO 2315
316         IP(N,1)=51
317         IND(N,1,2)=211
318         IND(N,1,3)=212
319 2315    DO 2320 K=1,KGROUP
320 2320    TH(K,N,1)=(DA(K)+DB(K))/2.0
321 2345    IND(N,1,4)=120
322         IND(N,1,5)=0
323 2350    CONTINUE
324         IF(IP(N,MCOL).EQ.40) GOTO 2480
325         IF(IP(N,MCOL).EQ.50) GOTO 2475
326         JA=JC(N-1,MCOL1)
327         JB=JC(N,MCOL1)
328         IF(JA ) 2370,2370,2390
329 2370    DO 2380 K=1,KGROUP
330 2380    DA(K)=DI(K,-JA)

```

```

331      GOTO 2400
332 2390 NVA=NVMAX(JA  )+1
333      MVA=MVMAX(JA  )+1
334      JA=JCV(JA,NVA,MVA)
335      DO 2395 K=1,KGROUP
336 2395 DA(K)=DI (K,JA)
337 2400 IF(JB  ) 2410,2410,2430
338 2410 DO 2420 K=1,KGROUP
339 2420 DB(K)=DI(K,-JB)
340      GOTO 2450
341 2430 MVA=MVMAX(JB  )+1
342      JB=JCV(JB,1,MVA)
343      DO 2440 K=1,KGROUP
344 2440 DB(K)=DI (K,JB)
345 2450 IF(IP(N,MCOL).EQ.41.OR.IP(N,MCOL).EQ.51) GOTO 2475
346      IF(IP(N,MCOL).EQ.421) GOTO 2465
347      IF(ABS(CA(KGROUP)-DB(KGROUP)).GT.1.0E-04.AND.BR.GT.0.5) GOTO 2460
348      IP(N,MCOL)=40
349      GOTO 2480
350 2460 IP(N,MCOL)=421
351      IF(JA.LT.0.AND.JB.LT.0) GOTO 2465
352      IP(N,MCOL)=51
353      IND(N,MCOL,2)=211
354      IND(N,MCOL,3)=212
355 2465 DO 2470 K=1,KGROUP
356 2470 TH(K,N,MCOL)=(DA(K)+DB(K))/2.0
357 2475 IND(N,MCOL,4)=120
358      IND(N,MCOL,5)=0
359 2480 CCONTINUE
360 2500 CONTINUE
361      DO 2560 M=1,MCOL
362      IF(BT-0.5)2510,2510,2520
363 2510 IND(1,M,1)=0
364      IND(1,M,3)=0
365      IND(1,M,4)=0
366      IND(1,M,5)=0
367      GOTO 2530
368 2520 IND(1,M,2)=0
369      IND(1,M,3)=0
370      IND(1,M,6)=0
371 2530 IF(BB-0.5)2540,2540,2550
372 2540 IND(NROW,M,1)=0
373      IND(NROW,M,3)=0
374      IND(NROW,M,4)=0
375      IND(NROW,M,5)=0
376      GOTO 2560
377 2550 IND(NROW,M,2)=0
378      IND(NROW,M,3)=0
379      IND(NROW,M,6)=0
380 2560 CONTINUE
381      DO 2620 N=1,NROW
382      IF(BL-0.5)2570,2570,2580
383 2570 IND(N,1,1)=0
384      IND(N,1,2)=0
385      IND(N,1,3)=0
386      IND(N,1,5)=0

```

```

387      GOTO 2590
388 2580 IND(N,1,4)=0
389      IND(N,1,5)=0
390      INC(N,1,6)=0
391 2590 IF(BR-0.5)2600,2600,2610
392 2600 IND(N,MCOL,1)=0
393      IND(N,MCOL,2)=0
394      IND(N,MCOL,3)=0
395      IND(N,MCOL,5)=0
396      GOTO 2620
397 2610 IND(N,MCOL,4)=0
398      INC(N,MCOL,5)=0
399      IND(N,MCOL,6)=0
400 2620 CONTINUE
401      L=0
402      DO 2800 N=1,NROW
403      DO 2800 M=1,MCOL
404      DO 2800 I=1,6
405      IF(IND(N,M,I).LE.0) GOTO 2800
406      L=L+1
407      IND(N,M,I)=IND(N,M,I)+1000*L
408 2800 CONTINUE
409      LMAX=L
410      LMAX1=LMAX-1
411      L1=1
412      L2=L1+KGROUP*LMAX
413      L3=L2+ KGROUP*LMAX
414      L4=L3+ K1*LMAX
415      CALL COEFF(LXIN,NROW,NROW1,MCOL,MCOL1,KGROUP,K1,JMAX,
1JVMAX,NVN,MVM,LMAX,LXMAX,
1LXC MAX,LXFMAX,A,AR(L1),AR(L2),AR(L3),AR(L4),F1,F,S,DI,SR,SF,ST,HN,
2HM,HNV,HMV,DA,DAA,DB,DBB,D1,D2,
3      IND,TH,LXD,LXDF,LLX,LLXF,JC,JCV,NVMAX,MVMAX,IP)
416      CALL CHLSKY(LXIN,EP SCH,KGROUP,LMAX,LXMAX,LXC MAX,A,AR(L1),AR(L3),
1LXD,LXDC,LLX,LLXC)
417      CALL APRINT(MSIZE,MSIZE1,NRUN1,ICMAX,NCCNV,EP SF,EP SL,EP SCH,BL,BR,
1BT,BB,JMAX,XNORM,NROW,MCOL,KGROUP,LMAX,LXMAX,LXC MAX,LXFMAX,
2JVMAX,NVN,MVM,IND,IP)
418      CALL BPRINT(NROW,NROW1,MCOL,MCOL1,KGROUP,K1,JMAX,JVMAX,NVN,MVM,
1      AR(L1),XI,DI,SR,SF,ST,HN,HM,HNV,HMV,JC,JCV,NVMAX,MVMAX)
419      L3=L2+LMAX
420      L4=L3+LMAX
421      CALL OUTIT(NP,IOUT,NCONV,EP SF,EP SL,NROW,MCOL,KGROUP,K1,
1LMAX,LXMAX,
1LXC MAX,LXFMAX,A,AR(L1),AR(L2),AR(L3),AR(L4),F1,F,S,IND,LXD,LXDC,
2LXDF,LLX,LLXC,LLXF)
422      L=LXMAX/2
423      LC=LXC MAX/2
424      LF=LXFMAX/2
425      LS=KGROUP*NROW1*MCOL1+5
426      IC MAX1=IC MAX
427      IF(IC MAX.LE.0) IC MAX1=1
428      CALL EXPAN(IC MAX,IC MAX1,XNORM,NROW,NROW1,MCOL,MCOL1,KGROUP,K1,
1JMAX,JVMAX,NVN,MVM,LMAX,ICLPS,
1      A(1,1),AR(L1),AR(L2),AR(L3),AR(L4),F1(1),F1(L),F(1,1),
2F(1,LF),S(1,1),S(1,LS),XI,DI,SR,SF,ST,HN,HM,HNV,HMV,

```

429
430

3 IND, TH, JC, JCV, NV MAX, MV MAX, LXD, LXDC, LXDF, LLX, LLXC, IP)
RETURN
END

```

431     SUBROUTINE COEFF(LXIN,NROW,NROW1,MCOL,MCCL1,KGROUP,K1,JMAX,
1JVMAX,NVN,MVM,LMAX,
1LXMAX,LXCMAX,LXFMAX,A,AA,AF,AS,IFLAG,F1,F,S,DI,SR,SF,ST,HN,HM,
2HNV,HMV,D1,D2,TDN1,TDN2,TDM1,TDM2,
2IND,TH,LXD,LXDF,LLX,LLXF,JC,JCV,NVMAX,MVMAX,IP)
432     REAL*8 A(KGROUP,LXCMAX),AA(KGROUP,LMAX),AF(KGROUP,LMAX),
1AS(K1,LMAX)
433     INTEGER*2 LXD(LMAX),LXDF(LMAX),LLX(LXMAX),LLXF(LXFMAX),
1JC(NROW1,MCOL1)
434     INTEGER*2 JCV(JVMAX,NVN,MVM),NVMAX(JVMAX),MVMAX(JVMAX)
435     INTEGER*2 IP(NROW,MCOL)
436     COMMON/COM11/L,LL,NI,NNI,MI,MMI,NS,NNS,MS,MMS,NA,MA,LLMIN,LLMAX
437     DIMENSION DI(KGROUP,JMAX),SR(KGROUP,JMAX),SF(KGROUP,JMAX),
1ST(K1,JMAX)
438     DIMENSION HN(NROW1),HM(MCOL1)
439     DIMENSION IND(NROW,MCOL,6),TH(KGROUP,NROW,MCOL)
440     DIMENSION F1(LXMAX),F(K1,LXFMAX),S(K1,LXFMAX)
441     DIMENSION IFLAG(LMAX)
442     DIMENSION HNV(JVMAX,NVN),HMV(JVMAX,MVM),D1(KGROUP),D2(KGROUP),
1TDN1(KGROUP),TDN2(KGROUP),TDM1(KGROUP),TDM2(KGROUP)
443     DO 1000 LL=1,LMAX
444     IFLAG(LL)=0
445     DO 1000 K=1,KGROUP
446     AA(K,LL)=0.0
447     AF(K,LL)=0.0
448     1000 IF(K.LE.K1) AS(K,LL)=0.0
449     LX=0
450     LXF=0
451     DO 1500 N=1,NROW
452     DO 1500 M=1,MCOL
453     DO 1500 I=1,6
454     L=IND(N,M,I)/1000
455     IF(L.LE.0) GOTO 1500
456     NI=(IND(N,M,I)-1000*L)/100
457     MI=(IND(N,M,I)-1000*L-100*NI)/10
458     NMIN=N-1
459     NMAX=N+1
460     IF(NMIN.LT.1) NMIN=1
461     IF(NMAX.GT.NROW) NMAX=NROW
462     MMIN=M-1
463     MMAX=M+1
464     IF(MMIN.LT.1) MMIN=1
465     IF(MMAX.GT.MCOL) MMAX=MCOL
466     NSMIN=1
467     NSMAX=2
468     MSMIN=1
469     MSMAX=2
470     IS=IND(N,M,I)-1000*L-100*NI-10*MI
471     IF(IS.GT.0.AND.NI.EQ.2) GOTO 1010
472     IF(IS.GT.0.AND.MI.EQ.2) GOTO 1020
473     GOTO 1030
474     1010 NSMIN=IS
475     NSMAX=IS
476     NMIN=N+(2*IS-3)
477     NMAX=NMIN
478     GOTO 1030

```

```

479 1020 MSMIN=IS
480      MSMAX=IS
481      MMIN=M+(2*IS-3)
482      MMAX=MMIN
483 1030 CONTINUE
484      LLMIN=L
485      LLMAX=1
486      DO 1180 NS=NSMIN,NSMAX
487      NB=N+(2*NS-3)
488      IF(NB.LT.1.OR.NB.GT.NROW) GOTO 1180
489      NA=N
490      IF(NB.GT.N)NA=NB
491      DO 1160 MS=MSMIN,MSMAX
492      NNS=NS
493      MMS=MS
494      MB=M+(2*MS-3)
495      IF(MB.LT.1.OR.MB.GT.MCOL) GOTO 1160
496      MA=M
497      IF(MB.GT.M)MA=MB
498      DO 1080 II=1,6
499      LL=IND(N,M,II)/1000
500      IF(LL.LE.0) GOTO 1080
501      NNI=(IND(N,M,II)-1000*LL)/100
502      MMI=(IND(N,M,II)-1000*LL-100*NNI)/10
503      IS=IND(N,M,II)-1000*LL-100*NNI-10*MMI
504      IF(IS.GT.0.AND.NNI.EQ.2.AND.IS.NE.NNS) GOTO 1080
505      IF(IS.GT.0.AND.MMI.EQ.2.AND.IS.NE.MMS) GOTO 1080
506      CALL MATRIX(N,M,I,N , M , II,NROW,NROW1,MCOL,MCOL1,KGROUP,K1,JMAX,
      1JVMAX,NVN,MVM,
      1 LMAX,AA,AF,AS,IFLAG,DI,SR,SF,ST,HN,HM,
      1 HNV,HMV,D1,D2,TDN1,TDN2,TDM1,TDM2,IND,TH,JC,JCV,NVMAX,MVMAX,IP)
507 1080 CONTINUE
508      NNS=(N-NB+3)/2
509      MMS=(M-MB+3)/2
510      DO 1140 II=1,6
511      LL=IND(NB,MB,II)/1000
512      IF(LL.LE.0) GOTO 1140
513      NNI=(IND(NB,MB,II)-1000*LL)/100
514      MMI=(IND(NB,MB,II)-1000*LL-100*NNI)/10
515      IS=IND(NB,MB,II)-1000*LL-100*NNI-10*MMI
516      IF(IS.GT.0.AND.NNI.EQ.2.AND.IS.NE.NNS) GOTO 1140
517      IF(IS.GT.0.AND.MMI.EQ.2.AND.IS.NE.MMS) GOTO 1140
518      CALL MATRIX(N,M,I,NB,MB,II,NROW,NROW1,MCOL,MCOL1,KGROUP,K1,JMAX,
      1JVMAX,NVN,MVM,
      1 LMAX,AA,AF,AS,IFLAG,DI,SR,SF,ST,HN,FM,
      1 HNV,HMV,D1,D2,TDN1,TDN2,TDM1,TDM2,IND,TH,JC,JCV,NVMAX,MVMAX,IP)
519 1140 CONTINUE
520 1160 CONTINUE
521 1180 CONTINUE
522      DO 1280 NS=NSMIN,NSMAX
523      NNS=NS
524      NB=N+(2*NS-3)
525      IF(NB.LT.1.OR.NB.GT.NROW) GOTO 1280
526      NA=N
527      IF(NB.GT.N)NA=NB
528      DO 1260 MM=MMIN,MMAX

```

```

529      IF(M.EQ.MM) GOTO 1260
530      MA=M
531      IF(MM.GT.M)MA=MM
532      MS=(MM-M+3)/2
533      MMS=(M-MM+3)/2
534      DO 1240 II=1,6
535      LL=IND(N,MM,II)/1000
536      IF(LL.LE.0) GOTO 1240
537      NNI=(IND(N,MM,II)-1000*LL)/100
538      MMI=(IND(N,MM,II)-1000*LL-100*NNI)/10
539      IS=IND(N,MM,II)-1000*LL-100*NNI-10*MMI
540      IF(IS.GT.0.AND.NNI.EQ.2.AND.IS.NE.NNS) GOTO 1240
541      IF(IS.GT.0.AND.MMI.EQ.2.AND.IS.NE.MMS) GOTO 1240
542      CALL MATRIX(N,M,I,N ,MM,II,NROW,NROW1,MCOL,MCOL1,KGROUP,K1,JMAX,
1JVMAX,NVN,MVM,
1      LMAX,AA,AF,AS,IFLAG,DI,SR,SF,ST,HN,HM,
1 HNV,HMV,D1,D2,TDN1,TDN2,TDM1,TDM2,IND,TH,JC,JCV,NVMAX,MVMAX,IP)
543 1240 CONTINUE
544 1260 CONTINUE
545 1280 CCONTINUE
546      DO 1440 MS=MSMIN,MSMAX
547      MMS=MS
548      MB=M+(2*MS-3)
549      IF(MB.LT.1.OR.MB.GT.MCOL) GOTO 1440
550      MA=M
551      IF(MB.GT.M)MA=MB
552      DO 1420 NN=NMIN,NMAX
553      IF(N.EQ.NN) GOTO 1420
554      NA=N
555      IF(NN.GT.N)NA=NN
556      NS=(NN-N+3)/2
557      NNS=(N-NN+3)/2
558      DO 1400 II=1,6
559      LL=IND(NN,M,II)/1000
560      IF(LL.LE.0) GOTO 1400
561      NNI=(IND(NN,M,II)-1000*LL)/100
562      MMI=(IND(NN,M,II)-1000*LL-100*NNI)/10
563      IS=IND(NN,M,II)-1000*LL-100*NNI-10*MMI
564      IF(IS.GT.0.AND.NNI.EQ.2.AND.IS.NE.NNS) GOTO 1400
565      IF(IS.GT.0.AND.MMI.EQ.2.AND.IS.NE.MMS) GOTO 1400
566      CALL MATRIX(N,M,I,NN,M ,II,NROW,NROW1,MCOL,MCOL1,KGROUP,K1,JMAX,
1JVMAX,NVN,MVM,
1      LMAX,AA,AF,AS,IFLAG,DI,SR,SF,ST,HN,HM,
1 HNV,HMV,D1,D2,TDN1,TDN2,TDM1,TDM2,IND,TH,JC,JCV,NVMAX,MVMAX,IP)
567 1400 CCONTINUE
568 1420 CONTINUE
569 1440 CCONTINUE
570      DO 1490 LL=LLMIN,L
571      IF(IFLAG(LL).LT.1) GOTO 1490
572      IFLAG(LL)=0
573      LX=LX+1
574      LXF=LXF+1
575      LLX(LX)=LL
576      LLXF(LXF)=LL
577      DO 1480 K=1,KGROUP
578      A(K,LX      )=AA(K,LL)

```

```

579      AA(K,LL)=0.0
580      IF(K-1) 1450,1450,1460
581 1450 F1(LX)=AF(K,LL)
582      GOTO 1470
583 1460 F(K-1,LXF)=AF(K,LL)
584 1470 AF(K,LL)=0.0
585      IF(K.EQ.KGROUP) GOTO 1480
586      S(K,LXF)=AS(K,LL)
587      AS(K,LL)=0.0
588 1480 CCONTINUE
589 1490 CONTINUE
590      LXD(L)=LX
591      IF(L.EQ.LMAX) GOTO 1498
592      L1=L+1
593      DO 1495 LL=L1,LLMAX
594      IF(IFLAG(LL).LT.1) GOTO 1495
595      IFLAG(LL)=0
596      LXF=LXF+1
597      LLXF(LXF)=LL
598      DO 1493 K=1,K1
599      F(K,LXF)=AF(K+1,LL)
600      AF(K+1,LL)=0.0
601      S(K,LXF)=AS(K,LL)
602 1493 AS(K,LL)=0.0
603 1495 CCONTINUE
604 1498 LXDF(L)=LXF
605 1500 CONTINUE
606      LXMAX=LX
607      LXFMAX=LXF
608      LXIN=LXCMAX-LXMAX
609      DO 1600 LXM=1,LXMAX
610      LX=LXMAX-LXM+1
611      DO 1600 K=1,KGROUP
612      A(K,LX+LXIN)=A(K,LX)
613 1600 A(K,LX)=0.0
614      RETURN
615      END

```

```

616     SUBROUTINE MATRIX(N,M,I,NN,MM,II,NROW,NRCW1,MCOL,MCOL1,KGROUP,K1,
        1JMAX,JVMAX,NVN,MVM,LMAX,AA,AF,AS,IFLAG,
        2DI,SR,SF,ST,HN,HM,HNV,HMV,D1,D2,TDN1,TDN2,TDM1,TDM2,
        3IND,TH,JC,JCV,NVMAX,MVMAX,IP)
617     REAL*8 AA(KGROUP,LMAX),AF(KGROUP,LMAX),AS(K1,LMAX)
618     REAL*8 HNV1,HNV2,HMV1,HMV2,XN,XDN,XM,XDM,XF,XD,XFV,XDV
619     INTEGER*2 JC(NROW1,MCOL1)
620     INTEGER*2 JCV(JVMAX,NVN,MVM),NVMAX(JVMAX),MVMAX(JVMAX)
621     INTEGER*2 IP(NROW,MCOL)
622     COMMON/COM11/L,LL,NI,NNI,MI,MMI,NS,NNS,MS,MMS,NA,MA,LLMIN,LLMAX
623     DIMENSION IFLAG(LMAX)
624     DIMENSION DI(KGROUP,JMAX),SR(KGROUP,JMAX),SF(KGROUP,JMAX),
        1ST(K1,JMAX)
625     DIMENSION HN(NROW1),HM(MCOL1)
626     DIMENSION IND(NROW,MCOL,6),TH(KGROUP,NROW,MCOL)
627     DIMENSION HNV(JVMAX,NVN),HMV(JVMAX,MVM),D1(KGROUP),D2(KGROUP),
        1TDN1(KGROUP),TDN2(KGROUP),TDM1(KGROUP),TDM2(KGROUP)
628     IF(LL.LT.LLMIN) LLMIN=LL
629     IF(LL.GT.LLMAX) LLMAX=LL
630     IFLAG(LL)=1
631     JA=JC(NA-1,MA-1)
632     IF(JA ) 900,900,920
633     900 DO 910 K=1,KGROUP
634         D1(K)=DI(K,-JA)
635     910 D2(K)=DI(K,-JA)
636         GOTO 940
637     920 NV1=1
638         NV2=1
639         MV1=1
640         MV2=1
641         NVMAX1=NVMAX(JA )+1
642         MVMAX1=MVMAX(JA )+1
643         IF((NA-1).LT.N) NV1=NVMAX1
644         IF((NA-1).LT.NN) NV2=NVMAX1
645         IF((MA-1).LT.M) MV1=MVMAX1
646         IF((MA-1).LT.MM) MV2=MVMAX1
647         JA1=JCV(JA,NV1,MV1)
648         JA2=JCV(JA,NV2,MV2)
649         DO 930 K=1,KGROUP
650             D1(K)=DI (K,JA1)
651     930 D2(K)=DI (K,JA2)
652     940 DO 1020 K=1,KGROUP
653         TDN1(K)=1.0
654         TDM1(K)=1.0
655         TDN2(K)=1.0
656         TDM2(K)=1.0
657         GOTO (1010,1002,1010,1004,1010,1006), I
658     1002 IF(IP(N,M).EQ.421) TDN1(K)=TH(K,N,M)/D1(K)
659         GOTO 1010
660     1004 IF(IP(N,M).EQ.422) TDM1(K)=TH(K,N,M)/D1(K)
661         GOTO 1010
662     1006 IF(TH(K,N,M).GT..0) TDN1(K)=TH(K,N,M)/D1(K)
663     1010 CONTINUE
664         GOTO (1020,1012,1020,1014,1020,1016), II
665     1012 IF(IP(NN,MM).EQ.421)
        2TDN2(K)=TH(K,NN,MM)/D2(K)

```

```

666      GOTO 1020
667 1014 IF(IP(NN,MM).EQ.422)
        2TDM2(K)=TH(K,NN,MM)/D2(K)
668      GOTO 1020
669 1016 IF(TH(K,NN,MM).GT..0)
        2TDN2(K)=TH(K,NN,MM)/D2(K)
670 1020 CONTINUE
671      DO 1100 K=1,KGROUP
672      IF(LL.GT.L) GOTO 1021
673      IF(JA   ) 10201,10201,10203
674 10201 XN=XF(NI,NS,NNI , NNS,TDN1(K),TDN2(K))*HN(NA-1)**(NI+NNI -1)
675      XDN=XD(NI,NS,NNI,NNS,TDN1(K),TDN2(K))*HN(NA-1)**(NI+NNI-3)
676      XM=XF(MI,MS,MMI , MMS,TDM1(K),TDM2(K))*HM(MA-1)**(MI+MMI -1)
677      XDM=XD(MI,MS,MMI,MMS,TDM1(K),TDM2(K))*HM(MA-1)**(MI+MMI-3)
678      AA(K,LL )=AA(K,LL)+ SR(K,-JA)*XN*XM+DI(K,-JA)*(XN*XDM+XM*XDN)
679      GOTO 1021
680 10203 HNV1=0.0
681      DO 10208 NV=1,NVMAX1
682      HNV2=HNV1+HNV(JA   ,NV)
683      HNV1=0.0
684      DO 10205 MV=1,MVMAX1
685      HNV2=HNV1+HNV(JA   ,MV)
686      XN =XFV(NI,NS,NNI,NNS,HN(NA-1),HNV1,HNV2,TDN1(K),TDN2(K))
687      XDN=XDV(NI,NS,NNI,NNS,HN(NA-1),HNV1,HNV2,TDN1(K),TDN2(K))
688      XM =XFV(MI,MS,MMI,MMS,HM(MA-1),HNV1,HNV2,TDM1(K),TDM2(K))
689      XDM=XDV(MI,MS,MMI,MMS,HM(MA-1),HNV1,HNV2,TDM1(K),TDM2(K))
690      JAV=JCV(JA,NV,MV)
691      AA(K,LL)=AA(K,LL)+SR (K,JAV   ) *XN*XM + DI (K,JAV   ) *
        1(XN*XDM+XM*XDN)
692 10205 HNV1=HNV2
693 10208 HNV1=HNV2
694 1021 IF(K.NE.1) GOTO 1050
695      IF(JA   ) 1031,1031,1033
696 1031 DO 1032 KK=1,KGROUP
697      IF(KK.EQ.1.AND.LL.GT.L) GOTO 1032
698      XN=XF(NI,NS,NNI , NNS,TDN1(K),TDN2(KK))*HN(NA-1)**(NI+NNI -1)
699      XM=XF(MI,MS,MMI , MMS,TDM1(K),TDM2(KK))*HM(MA-1)**(MI+MMI -1)
700      AF(KK,LL)=AF(KK,LL)+SF(KK,-JA)*XN*XM
701 1032 CCNTINUE
702      GOTO 1040
703 1033 DO 1039 KK=1,KGROUP
704      IF(KK.EQ.1.AND.LL.GT.L) GOTO 1039
705      HNV1=0.0
706      DO 1038 NV=1,NVMAX1
707      HNV2=HNV1+HNV(JA   ,NV)
708      HNV1=0.0
709      DO 1035 MV=1,MVMAX1
710      HNV2=HNV1+HNV(JA   ,MV)
711      JAV=JCV(JA,NV,MV)
712      IF(SF(KK,JAV).LT.1.0E-04) GOTO 1035
713      XN =XFV(NI,NS,NNI,NNS,HN(NA-1),HNV1,HNV2,TDN1(K),TDN2(KK))
714      XM =XFV(MI,MS,MMI,MMS,HM(MA-1),HNV1,HNV2,TDM1(K),TDM2(KK))
715      AF(KK,LL)=AF(KK,LL)+SF (KK,JAV   ) *XN*XM
716 1035 HNV1=HNV2
717 1038 HNV1=HNV2
718 1039 CONTINUE

```

```

719 1040 CONTINUE
720 GOTO 1100
721 1050 CONTINUE
722 IF(JA ) 1061,1061,1062
723 1061 XN=XF(NI,NS,NNI , NNS,TDN1(K),TDN2(K-1))*HN(NA-1)**(NI+NNI -1)
724 XM=XF(MI,MS,MMI , MMS,TDM1(K),TDM2(K-1))*HM(MA-1)**(MI+MMI -1)
725 AS(K-1,LL)=AS(K-1,LL)+ST(K-1,-JA)*XN*XM
726 GOTO 1100
727 1062 HNV1=0.0
728 DO 1068 NV=1,NVMAX1
729 HNV2=HNV1+HNV(JA ,NV)
730 HNV1=0.0
731 DO 1065 MV=1,MVMAX1
732 HNV2=HNV1+HNV(JA ,MV)
733 XN =XFV(NI,NS,NNI,NNS,HN(NA-1),HNV1,HNV2,TDN1(K),TDN2(K-1))
734 XM =XFV(MI,MS,MMI,MMS,HM(MA-1),HNV1,HNV2,TDM1(K),TDM2(K-1))
735 JAV=JCV(JA,NV,MV)
736 AS(K-1,LL)=AS(K-1,LL)+ST (K-1,JAV )*XN*XM
737 1065 HNV1=HNV2
738 1068 HNV1=HNV2
739 1100 CCNTINUE
740 RETURN
741 END

```

```

742      SUBROUTINE CHLSKY(LXIN, EPSCH, KGROUP, LMAX, LXMAX, LXCMAX, E, SUM, EAUX,
11XD, LXDC, LLX, LLXC)
743      REAL*8 E(KGROUP, LXCMAX)
744      REAL*8 SUM(KGROUP, LMAX), EAUX(KGROUP), DSQRT, DABS
745      INTEGER*2 LXD(LMAX), LXDC(LMAX), LLX(LXMAX), LLXC(LXCMAX)
746      DO 1050 K=1, KGROUP
747      DO 1050 LL=1, LMAX
748 1050  SUM(K, LL)=0.0
749      LXC=1
750      LXDC(1)=1
751      LLXC(1)=1
752      DO 1100 K=1, KGROUP
753 1100  E(K, 1)=DSQRT(E(K, LXIN+1))
754      DO 1500 L=2, LMAX
755      LXA=LXD(L-1)+2
756      LLMIN=LLX(LXA-1)
757      DO 1120 K=1, KGROUP
758      DO 1120 LL=LLMIN, L
759 1120  SUM(K, LL)=0.0
760      LXB=LXD(L)
761      LXCA=LXDC(L-1)+1
762      IFLAG=0
763      LL=LLX(LXA-1)
764      LXC1=LXDC(LL)
765      LXC=LXC+1
766      LLXC(LXC)=LL
767      DO 1110 K=1, KGROUP
768 1110  E(K, LXC)=E(K, LXA-1+LXIN)/E(K, LXC1)
769      DO 1500 LLX1=LXA, LXB
770      LLMAX=LLX(LLX1)
771      LLMIN=LLMIN+1
772      DO 1470 LL=LLMIN, LLMAX
773      IF(L-LL)1140, 1125, 1140
774 1125  DO 1130 LLLX1=LXCA, LXC
775      DO 1130 K=1, KGROUP
776      SUM(K, LL)=SUM(K, LL)+E(K, LLLX1)**2
777 1130  EAUX(K)=DSQRT(E(K, LXB+LXIN)-SUM(K, LL))
778      LXC=LXC+1
779      LXDC(L)=LXC
780      GOTO 1220
781 1140  LLXB=LXDC(LL)
782      LLXA=LXDC(LL-1)+1
783      IF(LLXC(LLXA).GT.LLXC(LXC)) GOTO 1165
784      DO 1160 LLLX1=LLXA, LLXB
785      LLL1=LLXC(LLX1)
786      IF(LLX1.GT.LLXC(LXC)) GOTO 1165
787      DO 1160 LLLX2=LXCA, LXC
788      IF(LLX1.NE.LLXC(LLX2)) GOTO 1160
789      DO 1150 K=1, KGROUP
790 1150  SUM(K, LL)=SUM(K, LL)+E(K, LLLX1)*E(K, LLLX2)
791 1160  CONTINUE
792 1165  IF(LL-LLMAX)1170, 1190, 1170
793 1170  DO 1180 K=1, KGROUP
794      EAUX(K)=-SUM(K, LL)/E(K, LLXB)
795 1180  IF(DABS(EAUX(K)).GT.EPSCH) IFLAG=1
796      GOTO 1210

```

```
797 1190 DO 1200 K=1,KGROUP
798     EAUX(K)=(E(K,LLX1+LXIN)-SUM(K,LL))/E(K,LLXB)
799 1200 IF(DABS(EAUX(K)).GT.EPSCHE) IFLAG=1
800 1210 IF(IFLAG.EQ.0) GOTO 1470
801     LXC=LXC+1
802 1220 LLXC(LXC)=LL
803     DO 1450 K=1,KGROUP
804 1450 E(K,LXC)=EAUX(K)
805 1470 CONTINUE
806     LLMIN=LLMAX
807 1500 CONTINUE
808     LXC=LXC+1
809     RETURN
810     END
```

```

811     SUBROUTINE OUTIT(NP,IOUT,NCONV,EPF,EP,L,NROW,MCOL,KGROUP,K1,LMAX,
      1LXMAX,LXCMAX,LXFMAX,E,Y,SS,YI,SUM,F1,F,S,IND,LXD,LXDC,LXDF,LLX,
      2LLXC,LLXF)
812     REAL*8 E(KGROUP,LXCMAX),Y(KGROUP,LMAX),SS(LMAX),YI(LMAX),SUM(LMAX)
813     REAL*8 YNUM,DEN,YLMD1,YLMD2,DABS
814     INTEGER*2 LXD(LMAX),LXDC(LMAX),LXDF(LMAX),LLX(LXMAX),LLXC(LXCMAX),
      1LLXF(LXFMAX)
815     DIMENSION F1(LXMAX),F(K1,LXFMAX),S(K1,LXFMAX)
816     DIMENSION IND(NROW,MCOL,6)
817     LMAX1=LMAX-1
818     IF(NP.GT.0) GOTO 1020
819     DO 1000 KK=1,KGROUP
820     DO 1000 LL=1,LMAX
821     1000 Y(KK,LL)=0.0
822     DO 1011 N=1,NROW
823     DO 1011 M=1,MCOL
824     LL=IND(N,M,1)/1000
825     IF(LL.LE.0) GOTO 1011
826     DO 1010 KK=1,KGROUP
827     1010 Y(KK,LL)=1.0
828     1011 CONTINUE
829     YLIM1=1.0
830     GOTO 1060
831     1020 YLIM1=0.0
832     DO 1030 KK=1,KGROUP
833     1030 READ(5,1040)(Y(KK,LL),LL=1,LMAX)
834     1040 FORMAT(6E12.5)
835     DO 1050 KK=1,KGROUP
836     DO 1050 LL=1,LMAX
837     1050 IF(YLIM1.LT.DABS(Y(KK,LL)))YLIM1=DABS(Y(KK,LL))
838     1060 YLMD1=1.0E 20
839     YLIM2=0.0
840     NCONV1=0
841     1100 NCONV1=NCONV1+1
842     IF(NCONV1.GT.NCONV) GOTO 1500
843     YNUM=0.0
844     DEN=0.0
845     DO 1300 K=1,KGROUP
846     DO 1110 L=1,LMAX
847     SS(L)=0.0
848     1110 YI( L)=Y(K,L)
849     IF(K.NE.1) GOTO 1210
C COMPUTATION OF FISSION SOURCE
850     DO 1150 L=1,LMAX
851     LX=LXD(L)
852     SS(L)=SS(L)+F1(LX)*Y(1,L)
853     IF(L.EQ.1) GOTO 1130
854     LX1=LXD(L-1)+1
855     LX2=LX-1
856     DO 1120 LX=LX1,LX2
857     LL=LLX(LX)
858     SS(L)=SS(L)+F1(LX)*Y(1,LL)
859     1120 SS(LL)=SS(LL)+F1(LX)*Y(1,L)
860     1130 LXF1=1
861     IF(L.GT.1) LXF1=LXDF(L-1)+1
862     LXF2=LXDF(L)

```

```

863      DO 1150 LXF=LXF1,LXF2
864      LL=LLXF(LXF)
865      DO 1150 KK=2,KGROUP
866      1150 SS(L)=SS(L)+F(KK-1,LXF)*Y(KK,LL)
867      GOTO 1250
C COMPUTATION OF SCATTERING SOURCE
868      1210 DO 1230 L=1,LMAX
869      LXF1=1
870      IF(L.GT.1) LXF1=LXDF(L-1)+1
871      LXF2=LXDF(L)
872      DO 1230 LXF=LXF1,LXF2
873      LL=LLXF(LXF)
874      1230 SS(L)=SS(L)+S(K-1,LXF)*Y(K-1,LL)
875      1250 Y(K,1)=SS(1)/E(K,1)
876      DO 1270 L=2,LMAX
877      SUM(L)=0.0
878      LXC1=LXDC(L-1)+1
879      LXC2=LXDC(L)-1
880      DO 1260 LXC=LXC1,LXC2
881      LL=LLXC(LXC)
882      1260 SUM(L)=SUM(L)+E(K,LXC)*Y(K,LL)
883      LXC=LXDC(L)
884      1270 Y(K,L)=(SS(L)-SUM(L))/E(K,LXC)
885      DO 1275 L=1,LMAX
886      SUM(L)=0.0
887      1275 SS(L)=Y(K,L)
888      Y(K,LMAX)=SS(LMAX)/E(K,LXC MAX)
889      DO 1285 LA=1,LMAX1
890      L=LMAX-LA+1
891      LXC1=LXDC(L-1)+1
892      LXC2=LXDC(L)-1
893      DO 1280 LXC=LXC1,LXC2
894      LL=LLXC(LXC)
895      1280 SUM(LL)=SUM(LL)+E(K,LXC)*Y(K,L)
896      LXC=LXDC(L-1)
897      Y(K,L-1)=(SS(L-1)-SUM(L-1))/E(K,LXC)
898      1285 CONTINUE
899      DO 1290 L=1,LMAX
900      YNUM=YNUM+Y(K,L)**2
901      1290 DEN=DEN+YI(L)*Y(K,L)
902      1300 CONTINUE
903      YLMD2=YNUM/DEN
904      DO 1400 K=1,KGROUP
905      DO 1400 L=1,LMAX
906      Y(K,L)=Y(K,L)/YLMD2
907      IF(YLIM2.LT.DABS(Y(K,L))) YLIM2=DABS(Y(K,L))
908      1400 CONTINUE
909      CALL CPRINT(NCONV1,YLIM1,YLIM2,YLMD1,YLMD2)
910      IF(ABS(YLIM2-YLIM1)/YLIM1.LT.EPSF.AND.DABS(YLMD2-YLMD1)/YLMD1
911      1.LT.EPSL) GOTO 1500
911      YLIM1=YLIM2
912      YLIM2=0.0
913      YLMD1=YLMD2
914      GOTO 1100
915      1500 IF(IOUT.LE.0) GOTO 1600
916      DO 1550 KK=1,KGROUP

```

```
917 1550 PUNCH 1040, (Y(KK,LL),LL=1,LMAX)  
918 1600 RETURN  
919 END
```

```

920     SUBROUTINE EXPAN(ICMAX,
1           ICMAX1,XNORM,NRCW,NROW1,MCOL,MCOL1,KGROUP,K1,
1JMAX,JVMAX,NVN,MVM,LMAX,
1ICLPS, X,Y,TDN,TDM,TD,AVS,AVA,AVF,AVD,XINT,FL,XI,DI,SR,SF,ST,HN,
1HM,HNV,HMV,IND,TH,JC,JCV,NVMAX,MVMAX,ISTR,ISTC,MINR,MINC,KC,IP)
921     REAL*8 Y(KGROUP,LMAX)
922     INTEGER*2 ISTR(ICMAX1),ISTC(ICMAX1),MINR(ICMAX1),MINC(ICMAX1),
1KC(ICMAX1),JC(NROW1,MCOL1)
923     INTEGER*2 JCV(JVMAX,NVN,MVM),NVMAX(JVMAX),MVMAX(JVMAX)
924     INTEGER*2 IP(NROW,MCOL)
925     DIMENSION X(KGROUP,NROW,MCOL,6)
926     DIMENSION TDN(2,2),TDM(2,2),TD(2,2)
927     DIMENSION DI(KGROUP,JMAX),SR(KGROUP,JMAX),SF(KGROUP,JMAX),
1ST(K1,JMAX),XI(KGROUP)
928     DIMENSION AVA(KGROUP,NROW1,MCOL1),AVF(KGROUP,NROW1,MCOL1),
1AVD(KGROUP,NROW1,MCOL1),AVS(KGROUP,NROW1,MCOL1)
929     DIMENSION HN(NROW1),HM(MCOL1)
930     DIMENSION IND(NROW,MCOL,6),TH(KGROUP,NROW,MCOL)
931     DIMENSION XINT(KGROUP,NROW1,MCOL1)
932     DIMENSION HNV(JVMAX,NVN),HMV(JVMAX,MVM)
933     DIMENSION FL(13,13)
934     IF(ICMAX.LE.0) GOTO 1005
935     DO 1003 J=1,ICMAX
936 1003 READ(5,1004)JJ,ISTR(JJ),ISTC(JJ),MINR(JJ),MINC(JJ),KC(JJ)
937 1004 FORMAT(14I5)
938 1005 CONTINUE
939     DO 1100 N=1,NROW
940     DO 1100 M=1,MCOL
941     DO 1100 I=1,6
942     L=IND(N,M,I)/1000
943     IF(L)1010,1010,1030
944 1010 DO 1020 K=1,KGROUP
945 1020 X(K,N,M,I)=0.0
946     GOTO 1100
947 1030 DO 1040 K=1,KGROUP
948 1040 X(K,N,M,I)=Y(K,L)
949 1100 CONTINUE
950     CALL DPRINT(NROW,MCOL,KGROUP,X,IND,IP)
951     FISS=0.0
952     DO 1205 N=1,NROW1
953     DO 1205 M=1,MCOL1
954     J=JC(N,M)
955     IF(J.LT.0) JA=-J
956     DO 1200 K=1,KGROUP
957     DO 1108 NI=1,2
958     NS=N+NI-1
959     IF(J.LT.0) GOTO 1103
960     NV=1
961     IF(NS.GT.N) NV=NVMAX(J)+1
962 1103 CONTINUE
963     DO 1108 MI=1,2
964     MS=M+MI-1
965     IF(J.LT.0) GOTO 1105
966     MV=1
967     IF(MS.GT.M) MV=MVMAX(J)+1
968     JA=JCV(J,NV,MV)

```

```

969 1105 CONTINUE
970   TDN(NI,MI)=1.0
971   TDM(NI,MI)=1.0
972   TD (NI,MI)=1.0
973   IF(IP(NS,MS).EQ.421)
1   TDN(NI,MI)=TH(K,NS,MS)/DI(K,JA)
974   IF(IP(NS,MS).EQ.422)
1   TDM(NI,MI)=TH(K,NS,MS)/DI(K,JA)
975   IF(TH(K,NS,MS).GT..0)
1   TD(NI,MI)=TH(K,NS,MS)/DI(K,JA)
976 1108 CONTINUE
977   XA21=X(K,N,M,2)
978   IF(IND(N,M,3).GT.0) XA21=X(K,N,M,3)
979   XB21=X(K,N,M+1,2)
980   IF(IND(N,M+1,3).GT.0) XB21=X(K,N,M+1,3)
981   XA12=X(K,N,M,4)
982   IF(IND(N,M,5).GT.0) XA12=X(K,N,M,5)
983   XC12=X(K,N+1,M,4)
984   IF(IND(N+1,M,5).GT.0) XC12=X(K,N+1,M,5)
985   XINT(K,N,M)=(X(K,N,M,1)+X(K,N,M+1,1)+X(K,N+1,M,1)+X(K,N+1,M+1,1))
1   *HN(N)*HM(M)/4. + (XA21*TDN(1,1)+XB21*TDN(1,2)-X(K,N+1,M,2)*TDN(2,
2   21)-X(K,N+1,M+1,2)*TDN(2,2))* (HN(N)**2*HM(M))/24. + (XA12*TDM(1,1)-
3   3X(K,N,M+1,4)*TDM(1,2)+XC12*TDM(2,1)-X(K,N+1,M+1,4)*TDM(2,2))* (HN
4   4(N)*HM(M)**2)/24. + (X(K,N,M,6)*TD(1,1)-X(K,N,M+1,6)*TD(1,2)-X(K,
5   5N+1,M,6)*TD(2,1)+X(K,N+1,M+1,6)*TD(2,2))* (HN(N)**2*HM(M)**2)/144.
986   IF(J)1110,1110,1140
987 1110 FISS=FISS+XINT(K,N,M)*SF(K,-J)/XI(K)
988   AVF(K,N,M)=          SF(K,-J)*XINT(K,N,M)
989   AVD(K,N,M)=          XINT(K,N,M)/DI(K,-J)
990   IF(K-KGROUP)1115,1120,1120
991 1115 SA=SR(K,-J)-ST(K,-J)
992   AVS(K,N,M)=ST(K,-J)*XINT(K,N,M)
993   GOTO 1125
994 1120 SA=SR(K,-J)
995 1125 AVA(K,N,M)=          SA*XINT(K,N,M)
996   GOTO 1200
997 1140 AVA(K,N,M)=0.0
998   AVF(K,N,M)=0.0
999   AVD(K,N,M)=0.0
1000   IF(K.LE.K1) AVS(K,N,M)=0.0
1001   NVMAX1=NVMAX(J)+1
1002   MVMAX1=MVMAX(J)+1
1003   HNV1=0.0
1004   DO 1190 NV=1,NVMAX1
1005   HNV2=HNV1+HNV(J,NV)
1006   U1N1 = 1./ (HN(N)**2)* (HNV2**3-HNV1**3)-1./ (2.*HN(N)**3)* (HNV2**4-
1   HNV1**4)
1007   U1N2 = -1./ (HN(N)**2)* ((HN(N)-HNV2)**3- (HN(N)-HNV1)**3)
1   +1./ (2.*HN(N)**3)* ((HN(N)-HNV2)**4- (HN(N)-HNV1)**4)
1008   U2N1 = -1./ (3.*HN(N))* (HNV2**3-HNV1**3)+1./ (4.*HN(N)**2)* (HNV2**4-
1   HNV1**4)
1009   U2N2 = -1./ (3.*HN(N))* ((HN(N)-HNV2)**3- (HN(N)-HNV1)**3)
1   +1./ (4.*HN(N)**2)* ((HN(N)-HNV2)**4- (HN(N)-HNV1)**4)
1010   HNV1=0.0
1011   DO 1180 MV=1,MVMAX1
1012   HNV2=HNV1+HNV(J,MV)

```

```

1013      U1M1 = 1./((HM(M)**2)*(HNV2**3-HMV1**3)-1./(2.*HM(M)**3)*(HNV2**4-
1          HMV1**4))
1014      U1M2 = -1./((HM(M)**2)*((HM(M)-HNV2)**3-(HM(M)-HNV1)**3)
1          +1./(2.*HM(M)**3)*((HM(M)-HNV2)**4-(HM(M)-HNV1)**4))
1015      U2M1 = -1./((3.*HM(M))*((HNV2**3-HMV1**3)+1./(4.*HM(M)**2)*(HNV2**4-
1          HMV1**4))
1016      U2M2 = -1./((3.*HM(M))*((HM(M)-HNV2)**3-(HM(M)-HNV1)**3)
1          +1./(4.*HM(M)**2)*((HM(M)-HNV2)**4-(HM(M)-HNV1)**4))
1017      T11=X(K,N,M,1)*U1N2*U1M2 + X(K,N,M+1,1)*U1N2*U1M1 + X(K,N+1,M,1)*
1          U1N1*U1M2 + X(K,N+1,M+1,1)*U1N1*U1M1
1018      T21=XA21*U2N2*U1M2*TDN(1,1)+XB21*U2N2*U1M1*TDN(1,2)+X(K,N+1,M,2)
1          *U2N1*U1M2*TDN(2,1)+X(K,N+1,M+1,2)*U2N1*U1M1*TDN(2,2)
1019      T12=XA12*U1N2*U2M2*TDM(1,1)+X(K,N,M+1,4)*U1N2*U2M1*TDM(1,2)
1          +XC12*U1N1*U2M2*TDM(2,1)+X(K,N+1,M+1,4)*U1N1*U2M1*TDM(2,2)
1020      T22=X(K,N,M,6)*U2N2*U2M2*TD(1,1)+X(K,N,M+1,6)*U2N2*U2M1*TD(1,2)
1          +X(K,N+1,M,6)*U2N1*U2M2*TD(2,1)+X(K,N+1,M+1,6)*U2N1*U2M1*TD(2,2)
1021      T=T11+T21+T12+T22
1022      JAV=JCV(J,NV,MV)
1023      FISS=FISS+T*SF(K,JAV)/XI(K)
1024      AVF(K,N,M)=AVF(K,N,M)+SF(K,JAV)*T
1025      AVD(K,N,M)=AVD(K,N,M)+T/DI(K,JAV)
1026      IF(K-KGROUP)1150,1160,1160
1027      1150 SA=SR(K,JAV)-ST(K,JAV)
1028      AVS(K,N,M)=AVS(K,N,M)+ST(K,JAV)*T
1029      GOTO 1170
1030      1160 SA=SR(K,JAV)
1031      1170 AVA(K,N,M)=AVA(K,N,M)+T*SA
1032      1180 HMV1=HNV2
1033      1190 HNV1=HNV2
1034      1200 CONTINUE
1035      1205 CONTINUE
1036      XNORM=XNORM/FISS
1037      DO 1208 N=1,NROW1
1038      DO 1208 M=1,MCOL1
1039      DO 1208 K=1,KGROUP
1040      AVA(K,N,M)=AVA(K,N,M)*XNORM
1041      AVF(K,N,M)=AVF(K,N,M)*XNORM
1042      AVD(K,N,M)=AVD(K,N,M)*XNORM
1043      IF(K.LE.K1) AVS(K,N,M)=AVS(K,N,M)*XNORM
1044      1208 XINT(K,N,M)=XINT(K,N,M)*XNORM/(HN(N)*HM(M))
1045      CALL EPRINT(NROW,NROW1,MCOL,MCOL1,KGROUP,K1,ICLPS,AVA,AVF,AVD,
1          IAVS,XINT,HN,HM,JC)
1046      JJ=0
1047      IF(ICMAX)1210,1600,1550
1048      1210 INMAX=-ICMAX+2
1049      IMMAX=-ICMAX+2
1050      DO 1500 K=1,KGROUP
1051      DO 1500 N=1,NROW1
1052      DO 1500 M=1,MCOL1
1053      JA=JC(N,M)
1054      JJ=JJ+1
1055      1500 CALL CALC(XNORM,NROW,MCOL,KGROUP,JMAX,
1          JVMAX, NVN,MVM,JA,JJ,INMAX,IMMAX,
1          IK,N,M,HN(N),HM(M),X,TDN,TDM,TD,IND,TH,FL,DI,NVMAX,MVMAX,JCV,IP)
1056      GOTO 1600
1057      1550 DO 1590 K=1,KGROUP

```

```

1058      DO 1590 N=1,NROW1
1059      DO 1590 M=1,MCOL1
1060      JA=JC(N,M)
1061      INMAX=5
1062      IMMAX=5
1063      DO 1560 JB=1,ICMAX
1064      IF(MINR(JB).EQ.N.AND.MINC(JB).EQ.M.AND.KC(JB).EQ.K) GOTO 1570
1065 1560 CONTINUE
1066      GOTO 1580
1067 1570 INMAX=ISTR(JB)+2
1068      IMMAX=ISTC(JB)+2
1069 1580 JJ=JJ+1
1070 1590 CALL CALC(XNORM,NROW,MCOL,KGROUP,JMAX,
      1          JVMAX,      NVN,MVM,JA,JJ,INMAX,IMMAX,
      1K,N,M,HN(N),HM(M),X,TDN,TDM,TD,IND,TH,FL,DI,NVMAX,MVMAX,JCV,IP)
1071 1600 RETURN
1072      END

```

```

1073      SUBROUTINE CALC(XNORM,NROW,MCOL,KGROUP,JMAX,
1          JVMAX,NVN,MVM,JA,JJ,INMAX,
1074      IMMAX,K,N,M,HNN,HMM,X,TDN,TDM,TD,IND,TH,FL,DI,NVMAX,MVMAX,JCV,IP)
1075      INTEGER*2 JCV(JVMAX,NVN,MVM),NVMAX(JVMAX),MVMAX(JVMAX)
1076      INTEGER*2 IP(NROW,MCOL)
1077      DIMENSION DI(KGROUP,JMAX)
1078      DIMENSION X(KGROUP,NROW,MCOL,6)
1079      DIMENSION TDN(2,2),TDM(2,2),TD(2,2)
1080      DIMENSION IND(NROW,MCOL,6),TH(KGROUP,NROW,MCOL)
1081      DIMENSION FL(13,13)
1082      J=10000*K+100*N+M
1083      IF(JA.LT.0) DA=DI(K,-JA)
1084      DO 1108 NI=1,2
1085      NS=N+NI-1
1086      IF(JA.LT.0) GOTO 1103
1087      NV=1
1088      IF(NS.GT.N) NV=NVMAX(JA)+1
1103 CONTINUE
1089      DO 1108 MI=1,2
1090      MS=M+MI-1
1091      IF(JA.LT.0) GOTO 1105
1092      MV=1
1093      IF(MS.GT.M) MV=MVMAX(JA)+1
1094      JAV=JCV(JA,NV,MV)
1095      DA=DI(K,JAV)
1105 CONTINUE
1096      TDN(NI,MI)=1.0
1097      TDM(NI,MI)=1.0
1098      TD(NI,MI)=1.0
1099      IF(IP(NS,MS).EQ.421)
1100      1TDN(NI,MI)=TH(K,NS,MS)/DA
1101      IF(IP(NS,MS).EQ.422)
1102      1TDM(NI,MI)=TH(K,NS,MS)/DA
1103      IF(TH(K,NS,MS).GT..0)
1104      1TD(NI,MI)=TH(K,NS,MS)/DA
1105 CONTINUE
1106      XA21=X(K,N,M,2)
1107      IF(IND(N,M,3).GT.0) XA21=X(K,N,M,3)
1108      XB21=X(K,N,M+1,2)
1109      IF(IND(N,M+1,3).GT.0) XB21=X(K,N,M+1,3)
1110      XA12=X(K,N,M,4)
1111      IF(IND(N,M,5).GT.0) XA12=X(K,N,M,5)
1112      XC12=X(K,N+1,M,4)
1113      IF(IND(N+1,M,5).GT.0) XC12=X(K,N+1,M,5)
1114      DO 1450 IN=1,INMAX
1115      Z=HNN*(IN-1)/(INMAX-1)
1116      H=HNN
1117      U1N1=3*(Z/H)**2-2*(Z/H)**3
1118      U1N2=3*((H-Z)/H)**2-2*((H-Z)/H)**3
1119      U2N1=(-(Z/H)**2+(Z/H)**3)*H
1120      U2N2=((H-Z)/H)**2-((H-Z)/H)**3)*H
1121      DO 1450 IM=1,IMMAX
1122      Z=HMM*(IM-1)/(IMMAX-1)
1123      H=HMM
1124      U1M1=3*(Z/H)**2-2*(Z/H)**3
1125      U1M2=3*((H-Z)/H)**2-2*((H-Z)/H)**3

```

```

1124      U2M1=(-(Z/H)**2+(Z/H)**3)*H
1125      U2M2=((H-Z)/H)**2-((H-Z)/H)**3)*H
1126      T11=X(K,N,M,1)*U1N2*U1M2 + X(K,N,M+1,1)*U1N2*U1M1 + X(K,N+1,M,1)*
1127      1U1N1*U1M2 + X(K,N+1,M+1,1)*U1N1*U1M1
1128      T21=XA21*U2N2*U1M2*TDN(1,1)+XB21*U2N2*U1M1*TDN(1,2)+X(K,N+1,M,2)
1129      1*U2N1*U1M2*TDN(2,1)+X(K,N+1,M+1,2)*U2N1*U1M1*TDN(2,2)
1130      T12=XA12*U1N2*U2M2*TDM(1,1)+X(K,N,M+1,4)*U1N2*U2M1*TDM(1,2)
1131      1+XC12*U1N1*U2M2*TDM(2,1)+X(K,N+1,M+1,4)*U1N1*U2M1*TDM(2,2)
1132      T22=X(K,N,M,6)*U2N2*U2M2*TD(1,1)+X(K,N,M+1,6)*U2N2*U2M1*TD(1,2)
1133      1+X(K,N+1,M,6)*U2N1*U2M2*TD(2,1)+X(K,N+1,M+1,6)*U2N1*U2M1*TD(2,2)
1134      FL(IN,IM)=(T11+T21+T12+T22)*XNORM
1450 CONTINUE
      CALL FPRINT(J,JJ,INMAX,IMMAX,FL)
      RETURN
      END

```

```

1135      SUBROUTINE APRINT(MSIZE,MSIZE1, NRUN1,ICMAX,NCONV, EPSF,EPST,EPSTCH,
1BL,BR,BT,BB,JMAX,XNORM,NROW,MCOL,KGROUP,LMAX,LXMAX,LXCMA, LXFMA,
2JVMAX,NVN,MVM,IND,IP)
1136      INTEGER*2 IP(NROW,MCOL)
1137      DIMENSION IND(NROW,MCOL,6)
1138      WRITE(6,1000)
1139      1000 FORMAT(1H1,          20X,'PROBLEM PARAMETERS')
1140      WRITE(6,1010)
1141      1010 FORMAT(21X,18('-',),//)
1142      WRITE(6,1015) NRUN1
1143      1015 FORMAT(5X,'RUN NUMBER           -',I3)
1144      WRITE(6,1020) NROW
1145      1020 FORMAT(5X,'NO. OF ROWS          -',I3)
1146      WRITE(6,1030) MCOL
1147      1030 FORMAT(5X,'NO. OF COLUMNS      -',I3)
1148      WRITE(6,1040) KGROUP
1149      1040 FORMAT(5X,'NO. OF GROUPS        -',I3)
1150      WRITE(6,1060) NCCNV
1151      1060 FORMAT(/,5X,'OUTER.ITER. - MAX.NO.      -',I3)
1152      WRITE(6,1070) EPSF
1153      1070 FORMAT(16X,' - CONV.CRIT.SOL.      -',E10.3)
1154      WRITE(6,1075) EPST
1155      1075 FORMAT(16X,' - CONV.CRIT.K-EFF      -',E10.3)
1156      WRITE(6,1078) EPSTCH
1157      1078 FORMAT(/,5X,'CHOLESKI REJECTION CRIT.      -',E10.3)
1158      WRITE(6,1160)
1159      1160 FORMAT(/,5X,'BOUNDARIES(0.0-ZERO FLUX; 1.0-ZERO DERIV.)')
1160      WRITE(6,1170) BL
1161      1170 FORMAT(17X,'- LEFT BND.           -',F4.1)
1162      WRITE(6,1180) BR
1163      1180 FORMAT(17X,'- RIGHT BND.          -',F4.1)
1164      WRITE(6,1190) BT
1165      1190 FORMAT(17X,'- TOP BND.            -',F4.1)
1166      WRITE(6,1200) BB
1167      1200 FORMAT(17X,'- BOTTOM BND.          -',F4.1)
1168      WRITE(6,1202) XNORM
1169      1202 FORMAT(/, 5X,'NORMALIZATION(FISSIENS/SEC)      -',E12.5)
1170      WRITE(6,1210) LMAX
1171      1210 FORMAT(/,5X,'LMAX -NO.OF UNKNOWN(/GROUP)      -',I6)
1172      WRITE(6,1215) LXMAX
1173      1215 FORMAT( 5X,'LXMAX -SIZE OF ARRAYS F1,LLX      -',I6)
1174      WRITE(6,1217) LXCMA
1175      1217 FORMAT( 5X,'LXCMA-SIZE OF ARRAYS E,LLXC      -',I6)
1176      WRITE(6,1230) LXFMA
1177      1230 FORMAT( 5X,'LXFMA-SIZE OF ARRAYS F,S,LLXF      -',I6)
1178      WRITE(6,1233) MSIZE
1179      1233 FORMAT(5X,'MSIZE -SPECIFIED VALUE      -',I8)
1180      WRITE(6,1235) MSIZE1
1181      1235 FORMAT(11X,'- ACTUAL VALUE USED      -',I8)
1182      AL=LMAX
1183      AK=KGROUP
1184      AX=LXMAX
1185      AXC=LXCMA
1186      AXF=LXFMA
1187      AJ=JMAX
1188      AJV=JVMAX

```

```

1189      ANV=NVN
1190      AMV=MVM
1191      AN=NRGW
1192      AM=MCOL
1193      MSIZE1=AL*(6.*AK+1.5)+AX*1.5+AXC*(2.*AK+0.5)+AXF*(2.*AK-1.5)
X+AJ*(4.*AK-1.0)+AJV*(ANV*AMV/2.0 +ANV+AMV+1.0)+AN*AM*(AK+7.0)
X+(AN+AM)/2.0+7.0*AK+4.0
1194      WRITE(6,1238) MSIZE1
1195 1238  FORMAT(11X,'-MINIMUM POSSIBLE VALUE  -',I8)
1196      WRITE(6,1240)
1197 1240  FORMAT(/,5X,'BASIS FUNCTIONS SPECIFICATION',/)
1198      DO 1260 N=1,NROW
1199 1260  WRITE(6,1270)(IP(N,M),M=1,MCOL)
1200 1270  FORMAT(15X,15I6)
1201      RETURN
1202      END

```

```

1203     SUBROUTINE BPRINT (NROW,NROW1,MCOL,MCOL1,KGROUP,K1,JMAX,JVMAX,NVN,
1 MVM,H,XI,DI,SR,SF,ST,HN,HM,HNV,HMV,JC,JCV,NVMAX,MVMAX)
1204     INTEGER*2 JC(NROW1,MCOL1)
1205     INTEGER*2 JCV(JVMAX,NVN,MVM),NVMAX(JVMAX),MVMAX(JVMAX)
1206     DIMENSION XI(KGROUP),DI(KGROUP,JMAX),SR(KGROUP,JMAX),SF(KGROUP,
1 JMAX),ST(K1,JMAX)
1207     DIMENSION HN(NROW1),HM(MCOL1),H(MCOL)
1208     DIMENSION HNV(JVMAX,NVN),HNV(JVMAX,MVM)
1209     WRITE(6,1000)
1210     1000 FORMAT(1H1,40X,'C O M P O S I T I O N   P R O P E R T I E S')
1211     WRITE(6,1010)
1212     1010 FORMAT(41X,43(' - '))
1213     WRITE(6,1020)
1214     1020 FORMAT(/,10X,'COMP.',3X,'GROUP',10X,'DIFF.COEFF(CM)',6X,'RMVL.XS(
1 /CM)',6X,'FISS.K->1(/CM)',5X,'SCTT.K->K+1(/CM)')
1215     WRITE(6,1030)
1216     1030 FORMAT(10X,5(' - '),3X,5(' - '),10X,14(' - '),6X,12(' - '),6X,14(' - '),5X,1
16(' - '))
1217     DO 1080 J=1,JMAX
1218     K=1
1219     WRITE(6,1040) J,K,DI(K,J),SR(K,J),SF(K,J),ST(K,J)
1220     1040 FORMAT(/,10X,I3,5X,I3,12X,F10.5,9X,F10.5,9X,F10.5,9X,F10.5)
1221     IF(K1.LE.1) GOTO 1065
1222     DO 1050 K=2,K1
1223     1050 WRITE(6,1060) K,DI(K,J),SR(K,J),SF(K,J),ST(K,J)
1224     1060 FORMAT(18X,
1 I3,12X,F10.5,9X,F10.5,9X,F10.5,9X,F10.5)
1225     1065 K=KGROUP
1226     WRITE(6,1070) K,DI(K,J),SR(K,J),SF(K,J)
1227     1070 FORMAT(18X,
1 I3,12X,F10.5,9X,F10.5,9X,F10.5)
1228     1080 CONTINUE
1229     WRITE(6,1090)(XI(K),K=1,KGROUP)
1230     1090 FORMAT(/,10X,'NEUTRON EMISSION FOR FISSIONS IN GROUP 1,-,KGROUP
1 -', 4F10.3)
1231     WRITE(6,1100)
1232     1100 FORMAT(1H1,40X,'M A T E R I A L S   P I C T U R E')
1233     WRITE(6,1110)
1234     1110 FORMAT(41X,33(' - '),/)
1235     H(1)=0.0
1236     DC 1111 M=2,MCOL
1237     1111 H(M)=H(M-1)+HM(M-1)
1238     WRITE(6,1115)(H(M),M=1,MCOL)
1239     1115 FORMAT(18X,12('(',F7.3,')',1X),/)
1240     WRITE(6,1120)(M,M=1,MCOL)
1241     1120 FORMAT(18X,12(I5,5X))
1242     DO 1121 N=2,NROW
1243     1121 H(N)=H(N-1)+HN(N-1)
1244     DC 1180 N=2,NROW
1245     WRITE(6,1130)(JC(N-1,M),M=1,MCOL1)
1246     1130 FORMAT(23X,12(I5,5X))
1247     WRITE(6,1140)H(N),N
1248     1140 FORMAT(11X,'(',F7.3,')',I3)
1249     1180 CONTINUE
1250     WRITE(6,1200)
1251     1200 FORMAT(1H1,30X,'SPECIFICATIONS FOR REGIONS WITH VARYING PARAMETERS
1 ')
1252     WRITE(6,1210)

```

```
1253 1210 FORMAT(31X,50('-'))
1254 DO 1500 J=1,JVMAX
1255 DO 1250 N=1,NROW1
1256 DO 1250 M=1,MCOL1
1257 IF(JC(N,M).EQ.J) GOTO 1300
1258 1250 CCNTINUE
1259 GOTO 1500
1260 1300 WRITE(6,1310) J
1261 1310 FORMAT(///,10X,'REGION NO.',I3)
1262 NVMAX1=NVMAX(J)+1
1263 MVMAX1=MVMAX(J)+1
1264 WRITE(6,1320)(HNV(J,NV),NV=1,NVMAX1)
1265 1320 FORMAT(/,10X,'ROW SUB-MESH',8F10.5)
1266 WRITE(6,1330)(HNV(J,MV),MV=1,MVMAX1)
1267 1330 FORMAT(/,10X,'COL.SUB-MESH',8F10.5)
1268 WRITE(6,1340)
1269 1340 FCRMAT(/,10X,'MATERIALS PICTURE OF SUB-MESH',/)
1270 DO 1400 NV=1,NVMAX1
1271 1400 WRITE(6,1410)(JCV(J,NV,MV),MV=1,MVMAX1)
1272 1410 FORMAT(20X,8I5)
1273 1500 CCNTINUE
1274 RETURN
1275 END
```

```
1276      SUBROUTINE CPRINT(NCONV1,YLIM1,YLIM2,YLMD1,YLMD2)
1277      REAL*8 YLMD1,YLMD2
1278      IF(NCONV1.NE.1) GOTO 1050
1279      WRITE(6,1000)
1280 1000  FORMAT(1H1,33X,'C O N V E R G E N C E   D A T A')
1281      WRITE(6,1010)
1282 1010  FORMAT(34X,31('-'),//)
1283      WRITE(6,1020)
1284 1020  FORMAT(10X,'OUT.ITER.NO.',3X,'INITIAL AND FINAL SOL. NORMS',4X,
1285      1'INITIAL AND FINAL K-EFF')
1285      WRITE(6,1030)
1286 1030  FORMAT(10X,12('-'),3X,28('-'),4X,23('-'))
1287 1050  WRITE(6,1060) NCONV1,YLIM1,YLIM2,YLMD1,YLMD2
1288 1060  FORMAT(10X,I6,4X,2(6X,F10.7,4X,F10.7))
1289      RETURN
1290      END
```

```

1291      SUBROUTINE DPRINT(NROW,MCOL,KGROUP,X,IND,IP)
1292      INTEGER*2 IP(NROW,MCOL)
1293      DIMENSION XX(5)
1294      DIMENSION X(KGROUP,NROW,MCOL,6)
1295      DIMENSION IND(NROW,MCOL,6)
1296      WRITE(6,1000)
1297 1000 FORMAT(1H1,30X,'E X P A N S I O N   C O E F F .   F O R   F L U X')
1298      WRITE(6,1010)
1299 1010 FORMAT(30X,49(' '),//)
1300      WRITE(6,1020)
1301 1020 FORMAT(5X,'GROUP',3X,'ROW',3X,'COLUMN',8X,'FLUX',11X,'COL.DER(UP)'
1302      1,5X,'COL.DER(DWN)',5X,'ROW DER(LFT)',5X,'ROW DER(RIT)',4X,'SECOND
1303      2DERIV.')
1304      WRITE(6,1030)
1305 1030 FORMAT(5X,5(' '),3X,3(' '),3X,6(' '),8X,4(' '),11X,11(' '),5X,
1306      112(' '),5X,12(' '),5X,12(' '),4X,13(' '))
1307      DO 1100 K=1,KGROUP
1308      DO 1100 N=1,NROW
1309      DO 1100 M=1,MCOL
1310      II=IP(N,M)/10
1311      IF(II.GT.10) II=II/10
1312      II=II-3
1313      GOTO (1034,1035,1036), II
1314 1036 WRITE(6,1120)K,N,M,(X(K,N,M,I),I=1,6)
1315 1120 FORMAT(5X,I3,4X,I3,5X,I3,2X,6(F15.7,2X))
1316      GOTO 1100
1317 1034 XX(1)=X(K,N,M,1)
1318      XX(2)=X(K,N,M,2)
1319      XX(3)=X(K,N,M,4)
1320      XX(4)=X(K,N,M,6)
1321      WRITE(6,1050)K,N,M,(XX(I),I=1,4)
1322 1050 FORMAT(5X,I3,4X,I3,5X,I3,2X,F15.7,2X,2( 9X,F15.7,10X),F15.7,2X)
1323      GOTO 1100
1324 1035 IF(IND(N,M,3).EQ.0) GOTO 1080
1325      XX(1)=X(K,N,M,1)
1326      XX(2)=X(K,N,M,2)
1327      XX(3)=X(K,N,M,3)
1328      XX(4)=X(K,N,M,4)
1329      XX(5)=X(K,N,M,6)
1330      WRITE(6,1070)K,N,M,(XX(I),I=1,5)
1331 1070 FORMAT(5X,I3,4X,I3,5X,I3,2X,3(F15.7,2X),9X,F15.7,10X,F15.7)
1332      GOTO 1100
1333 1080 XX(1)=X(K,N,M,1)
1334      XX(2)=X(K,N,M,2)
1335      XX(3)=X(K,N,M,4)
1336      XX(4)=X(K,N,M,5)
1337      XX(5)=X(K,N,M,6)
1338      WRITE(6,1090)K,N,M,(XX(I),I=1,5)
1339 1090 FORMAT(5X,I3,4X,I3,5X,I3,2X, F15.7,11X,F15.7,10X,3(F15.7,2X))
1340 1100 CONTINUE
1341      RETURN
1342      END

```

```

1340     SUBROUTINE EPRINT(NROW,NROW1,MCOL,MCOL1,KGROUP,K1,ICLPS,AVA,AVF,
1341     1AVD,AVS,XINT,HN,HM,JC)
1342     INTEGER*2 JC(NROW1,MCOL1)
1343     DIMENSION AVA(KGROUP,NROW1,MCOL1),AVF(KGROUP,NROW1,MCOL1),
1344     1AVD(KGROUP,NROW1,MCOL1),AVS(KGROUP,NROW1,MCOL1)
1345     DIMENSION XINT(KGROUP,NROW1,MCOL1),HN(NROW1),HM(MCOL1)
1346     DO 980 N=1,NROW1
1347     DO 980 M=1,MCOL1
1348     DO 980 K=2,KGROUP
1349     AVA(1,N,M)=AVA(1,N,M)+AVA(K,N,M)
1350     AVF(1,N,M)=AVF(1,N,M)+AVF(K,N,M)
1351     980 AVD(1,N,M)=AVD(1,N,M)+AVD(K,N,M)
1352     WRITE(6,1000)
1353     1000 FORMAT(1H1)
1354     WRITE(6,1010)
1355     1010 FORMAT(30X,'REACTION RATES IN REGIONS')
1356     WRITE(6,1020)
1357     1020 FORMAT(30X,25('-',),//)
1358     WRITE(6,1120)(M,M=1,MCOL)
1359     1120 FORMAT(8X,10(I6,6X))
1360     DO 1200 N=2,NROW
1361     WRITE(6,1130)(AVA(1,N-1,M),M=1,MCOL1)
1362     1130 FORMAT(6X,'ABSRP',2X,10E12.5)
1363     WRITE(6,1140)(AVF(1,N-1,M),M=1,MCOL1)
1364     1140 FORMAT(6X,'PRCDU',2X,10E12.5)
1365     WRITE(6,1150)(AVD(1,N-1,M),M=1,MCOL1)
1366     1150 FORMAT(6X,'TRNSP',2X,10E12.5)
1367     WRITE(6,1170)N
1368     1170 FORMAT(8X,I6)
1369     1200 CCNTINUE
1370     WRITE(6,1000)
1371     WRITE(6,2010)
1372     2010 FORMAT(30X,'AVERAGE FLUXES IN REGIONS')
1373     WRITE(6,2020)
1374     2020 FORMAT(30X,25('-',),//)
1375     WRITE(6,1120)(M,M=1,MCOL)
1376     DO 2200 N=2,NROW
1377     DO 2100 K=1,KGROUP
1378     WRITE(6,2130)K,(XINT(K,N-1,M),M=1,MCOL1)
1379     2130 FORMAT(6X,'GRP.',I1,2X,10E12.5)
1380     WRITE(6,1170) N
1381     2200 CONTINUE
1382     IF(ICLPS.EQ.0) GOTO 2500
1383     DO 2220 N=1,NROW1
1384     DO 2220 M=1,MCOL1
1385     DO 2220 K=2,KGROUP
1386     AVA(1,N,M)=AVA(1,N,M)-AVA(K,N,M)
1387     AVF(1,N,M)=AVF(1,N,M)-AVF(K,N,M)
1388     2220 AVD(1,N,M)=AVD(1,N,M)-AVD(K,N,M)
1389     WRITE(6,2230)
1390     2230 FORMAT(1H1,30X,'FLUX AND VOLUME AVERAGED PROPERTIES')
1391     WRITE(6,2240)
1392     2240 FORMAT(31X,35('-',),//)
1393     WRITE(6,2250)
1394     2250 FORMAT(10X,'ROW CCL',3X,'GROUP',10X,'DIFF.COEFF(CM)',6X,'RMVL.XS(/
1395     1CM)',6X,'FISS.K->1(/CM)',5X,'SCTT.K->K+1(/CM)')

```

```

1393      WRITE(6,2260)
1394 2260  FORMAT(10X,3(' '),1X,3(' '),3X,5(' '),10X,14(' '),6X,12(' '),6X,
        114(' '),5X,16(' '),/)
1395      DO 2275 N=1,NROW1
1396      DO 2275 M=1,MCOL1
1397      DO 2270 K=1,K1
1398      XX=XINT(K,N,M)*HN(N) *HM(M)
1399      AVA(K,N,M)=(AVA(K,N,M)+AVS(K,N,M))/XX
1400      AVS(K,N,M)=AVS(K,N,M)/XX
1401      AVF(K,N,M)=AVF(K,N,M)/XX
1402 2270  AVD(K,N,M)=XX/AVD(K,N,M)
1403      XX=XINT(KGROUP,N,M)*HN(N)*HM(M)
1404      AVA(KGROUP,N,M)=AVA(KGROUP,N,M)/XX
1405      AVF(KGROUP,N,M)=AVF(KGROUP,N,M)/XX
1406 2275  AVD(KGROUP,N,M)=XX/AVD(KGROUP,N,M)
1407      IF(ICLPS.LT.0) GOTO 2400
1408      DO 2340 NN=1,ICLPS
1409      READ(5,2280) N,M
1410 2280  FORMAT(14I5)
1411      K=1
1412      WRITE(6,2290) N,M,K,AVD(K,N,M),AVA(K,N,M),AVF(K,N,M),AVS(K,N,M)
1413 2290  FORMAT(/,8X,2I4,3X,I3,12X,F10.5,9X,F10.5,9X,F10.5,9X,F10.5)
1414      IF(K1.LE.1) GOTO 2330
1415      DO 2310 K=2,K1
1416 2310  WRITE(6,2320)      K,AVD(K,N,M),AVA(K,N,M),AVF(K,N,M),AVS(K,N,M)
1417 2320  FORMAT(19X,      I3,12X,F10.5,9X,F10.5,9X,F10.5,9X,F10.5)
1418 2330  K=KGROUP
1419      WRITE(6,2320)      K,AVD(K,N,M),AVA(K,N,M),AVF(K,N,M)
1420 2340  CCNTINUE
1421      GOTO 2500
1422 2400  CCNTINUE
1423      DO 2490 N=1,NROW1
1424      DO 2490 M=1,MCOL1
1425      IF(JC(N,M).LT.0) GOTO 2490
1426      K=1
1427      WRITE(6,2290) N,M,K,AVD(K,N,M),AVA(K,N,M),AVF(K,N,M),AVS(K,N,M)
1428      IF(K1.LE.1) GOTO 2430
1429      DO 2410 K=2,K1
1430 2410  WRITE(6,2320)      K,AVD(K,N,M),AVA(K,N,M),AVF(K,N,M),AVS(K,N,M)
1431 2430  K=KGROUP
1432      WRITE(6,2320)      K,AVD(K,N,M),AVA(K,N,M),AVF(K,N,M)
1433 2490  CONTINUE
1434 2500  CONTINUE
1435      RETURN
1436      END

```

```
1437     SUBROUTINE FPRINT(J,JJ,INMAX,IMMAX,FL)
1438     DIMENSION FL(13,13)
1439     IF(JJ.NE.1) GOTO 1100
1440     WRITE(6,1020)
1441 1020  FORMAT(1H1,      30X,'FLUX VALUES FOR EQUALLY SPACED POINTS INSIDE
1 THE OUTPUT REGIONS')
1442     WRITE(6,1030)
1443 1030  FORMAT(31X,63('-'))
1444     WRITE(6,1035)
1445 1035  FORMAT(50X,'(SET=GROUP-ROW-COLUMN)',//)
1446     WRITE(6,1040)
1447 1040  FORMAT(3X,'SET',1X,'SUB-ROW',20X,'POINT FLUXES AT SUB-COLUMN')
1448     WRITE(6,1050)
1449 1050  FORMAT(3X,3('-'),1X,7('-'),20X,26('-'),/)
1450     WRITE(6,1060)(JI,JI=1,13)
1451 1060  FORMAT(16X,13(I5,3X),/)
1452 1100  CONTINUE
1453     WRITE(6,1110) J
1454 1110  FORMAT(1X,I5)
1455     DO 1200 IN=1,INMAX
1456 1200  WRITE(6,1210)IN,(FL(IN,IM),IM=1,IMMAX)
1457 1210  FORMAT( 8X,I3,3X,13(F 8.4))
1458     RETURN
1459     END
```

```

1460     REAL FUNCTION XF*8 (JI,JS,JJI,JJS,TD1,TD2)
1461     REAL*8 P
1462     CCMCN/COMP/P(2,2,2,2)
1463     M=8*JI+4*JS+2*JJI+JJS-14
1464     GOTO (1111,1112,1121,1122,1211,1212,1221,1222,2111,2112,2121,
X2122,2211,2212,2221,2222), M
1465 1111 P(1,1,1,1) = 13./35.
1466     GOTO 5000
1467 1112 P(1,1,1,2) = 9./70.
1468     GOTO 5000
1469 1121 P(1,1,2,1) = -11./210.*TD2
1470     GOTO 5000
1471 1122 P(1,1,2,2) = 13./420.*TD2
1472     GOTO 5000
1473 1211 P(1,2,1,1) = 9./70.
1474     GOTO 5000
1475 1212 P(1,2,1,2) = 13./35.
1476     GOTO 5000
1477 1221 P(1,2,2,1) = -13./420.*TD2
1478     GOTO 5000
1479 1222 P(1,2,2,2) = 11./210.*TD2
1480     GOTO 5000
1481 2111 P(2,1,1,1) = -11./210.*TD1
1482     GOTO 5000
1483 2112 P(2,1,1,2) = -13./420.*TD1
1484     GOTO 5000
1485 2121 P(2,1,2,1) = 1./105.*TD1*TD2
1486     GOTO 5000
1487 2122 P(2,1,2,2) = -1./140.*TD1*TD2
1488     GOTO 5000
1489 2211 P(2,2,1,1) = 13./420.*TD1
1490     GOTO 5000
1491 2212 P(2,2,1,2) = 11./210.*TD1
1492     GOTO 5000
1493 2221 P(2,2,2,1) = -1./140.*TD1*TD2
1494     GOTO 5000
1495 2222 P(2,2,2,2) = 1./105.*TD1*TD2
1496 5000 XF=P(JI,JS,JJI,JJS)
1497     RETURN
1498     END

```

```

1499     REAL FUNCTION XD*8 (JI,JS,JJI,JJS,TD1,TD2)
1500     REAL*8 P
1501     COMMON/CCMP/P(2,2,2,2)
1502     M=8*JI+4*JS+2*JJI+JJS-14
1503     GOTO (1111,1112,1121,1122,1211,1212,1221,1222,2111,2112,2121,
        X2122,2211,2212,2221,2222), M
1504 1111 P(1,1,1,1) = 6./5.
1505     GOTO 5000
1506 1112 P(1,1,1,2) = -6./5.
1507     GOTO 5000
1508 1121 P(1,1,2,1) = -1./10.*TD2
1509     GOTO 5000
1510 1122 P(1,1,2,2) = -1./10.*TD2
1511     GOTO 5000
1512 1211 P(1,2,1,1) = -6./5.
1513     GOTO 5000
1514 1212 P(1,2,1,2) = 6./5.
1515     GOTO 5000
1516 1221 P(1,2,2,1) = 1./10.*TD2
1517     GOTO 5000
1518 1222 P(1,2,2,2) = 1./10.*TD2
1519     GOTO 5000
1520 2111 P(2,1,1,1) = -1./10.*TD1
1521     GOTO 5000
1522 2112 P(2,1,1,2) = 1./10.*TD1
1523     GOTO 5000
1524 2121 P(2,1,2,1) = 2./15.*TD1*TD2
1525     GOTO 5000
1526 2122 P(2,1,2,2) = -1./30.*TD1*TD2
1527     GOTO 5000
1528 2211 P(2,2,1,1) = -1./10.*TD1
1529     GOTO 5000
1530 2212 P(2,2,1,2) = 1./10.*TD1
1531     GOTO 5000
1532 2221 P(2,2,2,1) = -1./30.*TD1*TD2
1533     GOTO 5000
1534 2222 P(2,2,2,2) = 2./15.*TD1*TD2
1535 5000 XD=P(JI,JS,JJI,JJS)
1536     RETURN
1537     END

```

```

1538 REAL FUNCTION XFV*8(JI,JS,JJI,JJS,AH, X1, X2, TD1, TD2)
1539 REAL*8 H,X1,X2
1540 REAL*8 P
1541 CCOMMON/COMP/P(2,2,2,2)
1542 H=AH
1543 M=8*JI+4*JS+2*JJI+JJS-14
1544 GOTO (1111,1112,1121,1122,1211,1212,1221,1222,2111,2112,2121,
X2122,2211,2212,2221,2222), M
1545 1111 P(1,1,1,1)= +4./(7.*H**6)*(X2**7-X1**7)-2./H**5*(X2**6-X1**6)
2 +9./(5.*H**4)*(X2**5-X1**5)
1546 GOTO 5000
1547 1112 P(1,1,1,2)= -4./(7.*H**6)*(X2**7-X1**7)+2./H**5*(X2**6-X1**6)
2 -9./(5.*H**4)*(X2**5-X1**5)-1./(2.*H**3)*(X2**4-X1**4)
3 +1./H**2*(X2**3-X1**3)
1548 GOTO 5000
1549 1121 P(1,1,2,1)=(-2./(7.*H**5)*(X2**7-X1**7)+5./(6.*H**4)*(X2**6-X1**6)
2 -3./(5.*H**3)*(X2**5-X1**5))*TD2
1550 GOTO 5000
1551 1122 P(1,1,2,2)=(-2./(7.*H**5)*(X2**7-X1**7)+7./(6.*H**4)*(X2**6-X1**6)
2 -8./(5.*H**3)*(X2**5-X1**5)+3./(4.*H**2)*(X2**4-X1**4)
3 )*TD2
1552 GOTO 5000
1553 1211 P(1,2,1,1)= -4./(7.*H**6)*(X2**7-X1**7)+2./H**5*(X2**6-X1**6)
2 -9./(5.*H**4)*(X2**5-X1**5)-1./(2.*H**3)*(X2**4-X1**4)
3 +1./H**2*(X2**3-X1**3)
1554 GOTO 5000
1555 1212 P(1,2,1,2)= -4./(7.*H**6)*((H-X2)**7-(H-X1)**7)+2./H**5*((H-X2)**6
2 -(H-X1)**6)-9./(5.*H**4)*((H-X2)**5-(H-X1)**5)
1556 GOTO 5000
1557 1221 P(1,2,2,1)=(+2./(7.*H**5)*(X2**7-X1**7)-5./(6.*H**4)*(X2**6-X1**6)
2 +3./(5.*H**3)*(X2**5-X1**5)+1./(4.*H**2)*(X2**4-X1**4)
3 -1./(3.*H)*(X2**3-X1**3))*TD2
1558 GOTO 5000
1559 1222 P(1,2,2,2)=(-2./(7.*H**5)*((H-X2)**7-(H-X1)**7)+5./(6.*H**4)*
2 ((H-X2)**6-(H-X1)**6)-3./(5.*H**3)*((H-X2)**5-(H-X1)**
3 5))*TD2
1560 GOTO 5000
1561 2111 P(2,1,1,1)=(-2./(7.*H**5)*(X2**7-X1**7)+5./(6.*H**4)*(X2**6-X1**6)
2 -3./(5.*H**3)*(X2**5-X1**5))*TD1
1562 GOTO 5000
1563 2112 P(2,1,1,2)=(+2./(7.*H**5)*(X2**7-X1**7)-5./(6.*H**4)*(X2**6-X1**6)
2 +3./(5.*H**3)*(X2**5-X1**5)+1./(4.*H**2)*(X2**4-X1**4)
3 -1./(3.*H)*(X2**3-X1**3))*TD1
1564 GOTO 5000
1565 2121 P(2,1,2,1)=(+1./(7.*H**4)*(X2**7-X1**7)-1./(3.*H**3)*(X2**6-X1**6)
2 +1./(5.*H**2)*(X2**5-X1**5))*TD1*TD2
1566 GOTO 5000
1567 2122 P(2,1,2,2)=(+1./(7.*H**4)*(X2**7-X1**7)-1./(2.*H**3)*(X2**6-X1**6)
2 +3./(5.*H**2)*(X2**5-X1**5)-1./(4.*H)*(X2**4-X1**4)
3 )*TD1*TD2
1568 GOTO 5000
1569 2211 P(2,2,1,1)=(-2./(7.*H**5)*(X2**7-X1**7)+7./(6.*H**4)*(X2**6-X1**6)
2 -8./(5.*H**3)*(X2**5-X1**5)+3./(4.*H**2)*(X2**4-X1**4)
3 )*TD1
1570 GOTO 5000
1571 2212 P(2,2,1,2)=(-2./(7.*H**5)*((H-X2)**7-(H-X1)**7)+5./(6.*H**4)*

```

```

2          ((H-X2)**6-(H-X1)**6)-3./(5.*H**3)*((H-X2)**5-(H-X1)**
3          5))*TD1
1572      GOTO 5000
1573      2221 P(2,2,2,1)=(+1./(7.*H**4)*(X2**7-X1**7)-1./(2.*H**3)*(X2**6-X1**6)
2          +3./(5.*H**2)*(X2**5-X1**5)-1./(4.*H)*(X2**4-X1**4)
3          )*TD1*TD2
1574      GOTO 5000
1575      2222 P(2,2,2,2)=(-1./(7.*H**4)*((H-X2)**7-(H-X1)**7)+1./(3.*H**3)*((H-
2          X2)**6-(H-X1)**6)-1./(5.*H**2)*((H-X2)**5-(H-X1)**5)
3          )*TD1*TD2
1576      5000 XFV=P(JI,JS,JJI,JJS)
1577      RETURN
1578      END

```

```

1579     REAL FUNCTION XDV*8(JI,JS,JJI,JJS,AH, X1, X2, TD1, TD2)
1580     REAL*8 H,X1,X2
1581     REAL*8 P
1582     COMMON/CCMP/P(2,2,2,2)
1583     H=AH
1584     M=8*JI+4*JS+2*JJI+JJS-14
1585     GOTO (1111,1112,1121,1122,1211,1212,1221,1222,2111,2112,2121,
      X2122,2211,2212,2221,2222), M
1586 1111 P(1,1,1,1)= +36./(5.*H**6)*(X2**5-X1**5)-18./H**5*(X2**4-X1**4)
      2          +12./H**4*(X2**3-X1**3)
1587     GOTO 5000
1588 1112 P(1,1,1,2)= -36./(5.*H**6)*(X2**5-X1**5)+18./H**5*(X2**4-X1**4)
      2          -12./H**4*(X2**3-X1**3)
1589     GOTO 5000
1590 1121 P(1,1,2,1)=(-18./(5.*H**5)*(X2**5-X1**5)+15./(2.*H**4)*(X2**4-X1**
      2          4)-4./H**3*(X2**3-X1**3))*TD2
1591     GOTO 5000
1592 1122 P(1,1,2,2)=(-18./(5.*H**5)*(X2**5-X1**5)+21./(2.*H**4)*(X2**4-X1**
      2          4)-10./H**3*(X2**3-X1**3)+3./H**2*(X2**2-X1**2))*TD2
1593     GOTO 5000
1594 1211 P(1,2,1,1)= -36./(5.*H**6)*(X2**5-X1**5)+18./H**5*(X2**4-X1**4)
      2          -12./H**4*(X2**3-X1**3)
1595     GOTO 5000
1596 1212 P(1,2,1,2)= -36./(5.*H**6)*((H-X2)**5-(H-X1)**5)+18./H**5*((H-X2)**
      2          *4-(H-X1)**4)-12./H**4*((H-X2)**3-(H-X1)**3)
1597     GOTO 5000
1598 1221 P(1,2,2,1)=(+18./(5.*H**5)*(X2**5-X1**5)-15./(2.*H**4)*(X2**4-X1**
      2          4)+4./H**3*(X2**3-X1**3))*TD2
1599     GOTO 5000
1600 1222 P(1,2,2,2)=(-18./(5.*H**5)*((H-X2)**5-(H-X1)**5)+15./(2.*H**4)*
      2          ((H-X2)**4-(H-X1)**4)-4./H**3*((H-X2)**3-(H-X1)**3)
      3          )*TD2
1601     GOTO 5000
1602 2111 P(2,1,1,1)=(-18./(5.*H**5)*(X2**5-X1**5)+15./(2.*H**4)*(X2**4-X1**
      2          4)-4./H**3*(X2**3-X1**3))*TD1
1603     GOTO 5000
1604 2112 P(2,1,1,2)=(+18./(5.*H**5)*(X2**5-X1**5)-15./(2.*H**4)*(X2**4-X1**
      2          4)+4./H**3*(X2**3-X1**3))*TD1
1605     GOTO 5000
1606 2121 P(2,1,2,1)=(+9./(5.*H**4)*(X2**5-X1**5)-3./H**3*(X2**4-X1**4)
      2          +4./(3.*H**2)*(X2**3-X1**3))*TD1*TD2
1607     GOTO 5000
1608 2122 P(2,1,2,2)=(+9./(5.*H**4)*(X2**5-X1**5)-9./(2.*H**3)*(X2**4-X1**4)
      2          +11./(3.*H**2)*(X2**3-X1**3)-1./H*(X2**2-X1**2)
      3          )*TD1*TD2
1609     GOTO 5000
1610 2211 P(2,2,1,1)=(-18./(5.*H**5)*(X2**5-X1**5)+21./(2.*H**4)*(X2**4-X1**
      2          4)-10./H**3*(X2**3-X1**3)+3./H**2*(X2**2-X1**2))*TD1
1611     GOTO 5000
1612 2212 P(2,2,1,2)=(-18./(5.*H**5)*((H-X2)**5-(H-X1)**5)+15./(2.*H**4)*
      2          ((H-X2)**4-(H-X1)**4)-4./H**3*((H-X2)**3-(H-X1)**3)
      3          )*TD1
1613     GOTO 5000
1614 2221 P(2,2,2,1)=(+9./(5.*H**4)*(X2**5-X1**5)-9./(2.*H**3)*(X2**4-X1**4)
      2          +11./(3.*H**2)*(X2**3-X1**3)-1./H*(X2**2-X1**2)
      3          )*TD1*TD2

```

```
1615      GOTO 5000
1616      2222 P(2,2,2,2)=(-9./(5.*H**4)*((H-X2)**5-(H-X1)**5)+3./H**3*((H-X2)**4
          2      -(H-X1)**4)-4./(3.*H**2)*((H-X2)**3-(H-X1)**3)
          3      )*TD1*TD2
1617      5000 XDV=P(JI,JS,JJI,JJS)
1618      RETURN
1619      END
```

PROBLEM PARAMETERS

RUN NUMBER - 1
NO. OF ROWS - 3
NO. OF COLUMNS - 3
NO. OF GROUPS - 2

OUTER.ITER. - MAX.NO. - 50
- CONV.CRIT.SOL. - 0.100E-03
- CONV.CRIT.K-EFF - 0.100E-03

CHOLESKI REJECTION CRIT. - 0.100E-05

BOUNDARIES(0.0-ZERO FLUX; 1.0-ZERO DERIV.)
- LEFT BND. - 1.0
- RIGHT BND. - 0.0
- TOP BND. - 1.0
- BOTTOM BND. - 0.0

NORMALIZATION(FISSIONS/SEC) - 0.10000E 03

LMAX -NO.OF UNKNOWN(S)/GROUP - 20
LXMAX -SIZE OF ARRAYS F1,LLX - 133
LXCMAx-SIZE OF ARRAYS E,LLXC - 172
LXFMAx-SIZE OF ARRAYS F,S,LLXF - 246
MSIZE -SPECIFIED VALUE - 40000
-ACTUAL VALUE USED - 33576
-MINIMUM POSSIBLE VALUE - 2012

BASIS FUNCTIONS SPECIFICATION

40	51	40
51	61	40
40	40	40

C O M P O S I T I O N P R O P E R T I E S

<u>COMP.</u>	<u>GROUP</u>	<u>DIFF. COEFF (CM)</u>	<u>RMVL. XS (/CM)</u>	<u>FISS. K->1 (/CM)</u>	<u>SCTT. K->K+1 (/CM)</u>
1	1	1.20000	0.10100	0.00000	0.10000
	2	0.15000	0.02000	0.00000	
2	1	1.50000	0.06230	0.00000	0.06000
	2	0.40000	0.20000	0.21800	
3	1	1.55000	0.06230	0.00000	0.06000
	2	0.41330	0.25000	0.21800	
4	1	1.60000	0.06230	0.00000	0.06000
	2	0.42670	0.30000	0.21800	
5	1	1.65000	0.06230	0.00000	0.06000
	2	0.44000	0.35000	0.21800	

NEUTRON EMISSION FOR FISSIONS IN GROUP 1,--,KGROUP - 1.000 1.000

M A T E R I A L S P I C T U R E

	(0.000)	(20.000)	(40.000)	(
	1	2	3	
		1	-1	
(20.000)	2			
		-1	-1	
(40.000)	3			

SPECIFICATIONS FOR REGIONS WITH VARYING PARAMETERS

REGION NO. 1

ROW SUB-MESH 5.00000 5.00000 5.00000 5.00000

COL. SUB-MESH 5.00000 5.00000 5.00000 5.00000

MATERIALS PICTURE OF SUB-MESH

5	4	3	2
4	4	3	2
3	3	3	2
2	2	2	2

C O N V E R G E N C E D A T A

OUT.ITER.NO.	INITIAL AND FINAL SOL. NORMS		INITIAL AND FINAL K-EFF	
1	1.000000	1.7181350	*****	1.8955043
2	1.7181350	1.4432300	1.8955043	0.7529185
3	1.4432300	1.3020120	0.7529185	0.7410499
4	1.3020120	1.2286100	0.7410499	0.7399612
5	1.2286100	1.1901770	0.7399612	0.7406321
6	1.1901770	1.1698290	0.7406321	0.7413442
7	1.1698290	1.1589050	0.7413442	0.7418530
8	1.1589050	1.1529510	0.7418530	0.7421836
9	1.1529510	1.1496600	0.7421836	0.7423912
10	1.1496600	1.1478140	0.7423912	0.7425195
11	1.1478140	1.1467680	0.7425195	0.7425982
12	1.1467680	1.1461680	0.7425982	0.7426460
13	1.1461680	1.1458220	0.7426460	0.7426750
14	1.1458220	1.1456210	0.7426750	0.7426925
15	1.1456210	1.1455030	0.7426925	0.7427031
16	1.1455030	1.1454330	0.7427031	0.7427094

E X P A N S I O N C O E F F . F O R F L U X

C O L U M N	F L U X	C O L . D E R (U P)	C O L . D E R (D W N)	R O W D E R (L F T)	R O W D E R (R I G H T)
1	1.1454330		0.0000000		0.0000000
2	1.0562280		0.0000000	-0.1318337	-0.2378771
3	0.0000000		0.0000000		-0.0348431
1	1.0562280	-0.1318337	-0.2378771		0.0000000
2	0.5029880	-0.0780584	-0.1047543	-0.0780584	-0.1047543
3	0.0000000		0.0000000		-0.0144136
1	0.0000000		-0.0348431		0.0000000
2	0.0000000		-0.0144136		0.0000000
3	0.0000000		0.0000000		0.0000000
1	0.2573733		0.0000000		0.0000000
2	0.7405993		0.0000000	0.0756730	0.2156289
3	0.0000000		0.0000000		0.1251434
1	0.7405993	0.0756730	0.2156289		0.0000000
2	0.6186165	0.0164950	0.0268508	0.0164950	0.0268508
3	0.0000000		0.0000000		0.0439566
1	0.0000000		0.1251434		0.0000000
2	0.0000000		0.0439566		0.0000000
3	0.0000000		0.0000000		0.0000000

REACTION RATES IN REGIONS

	1		2		3
ABSRP	0.10998E	03	0.10450E	02	
PRODU	0.10000E	03	0.00000E	00	
TRNSP	0.21168E	04	0.35758E	04	
	2				
ABSRP	0.10450E	02	0.22135E	01	
PRODU	0.00000E	00	0.00000E	00	
TRNSP	0.35758E	04	0.74883E	03	
	3				

AVERAGE FLUXES IN REGIONS

	1	2	3
GRP.1	0.38355E 01	0.46288E 00	
GRP.2	0.11468E 01	0.12831E 01	
	2		
GRP.1	0.46288E 00	0.54992E-01	
GRP.2	0.12831E 01	0.27394E 00	
	3		

FLUX AND VOLUME AVERAGED PROPERTIES

<u>ROW</u>	<u>COL</u>	<u>GROUP</u>	<u>DIFF. COEFF (CM)</u>	<u>RMVL. XS (/CM)</u>	<u>FISS. K->1 (/CM)</u>	<u>SCTT. K->K+1 (/CM)</u>
1	1	1	1.54470	0.06230	0.00000	0.06000
		2	0.40824	0.23207	0.21800	

FLUX VALUES FOR EQUALLY SPACED POINTS INSIDE TH

(SET=GROUP-ROW-COLUMN)

SET SUB-ROW		POINT FLUXES AT SUB-COLUMN							
		1	2	3	4	5	6	7	8
10101	1	3.2921	3.5080	3.9294	4.2173	4.0324	3.0358		
	2	3.5080	3.7076	4.0872	4.3176	4.0696	3.0140		
	3	3.9294	4.0872	4.3620	4.4557	4.0700	2.9068		
	4	4.2173	4.3176	4.4557	4.3870	3.8671	2.6515		
	5	4.0324	4.0696	4.0700	3.8671	3.2944	2.1854		
	6	3.0358	3.0140	2.9068	2.6515	2.1854	1.4457		
10102	1	3.0358	1.0339	0.1904	0.0443	0.1345	0.0000		
	2	3.0140	1.0208	0.1811	0.0367	0.1291	0.0000		
	3	2.9068	0.9752	0.1609	0.0217	0.1155	0.0000		
	4	2.6515	0.8879	0.1412	0.0112	0.0975	0.0000		
	5	2.1854	0.7497	0.1335	0.0167	0.0789	0.0000		
	6	1.4457	0.5511	0.1492	0.0501	0.0637	0.0000		
10201	1	3.0358	3.0140	2.9068	2.6515	2.1854	1.4457		
	2	1.0339	1.0208	0.9752	0.8879	0.7497	0.5511		
	3	0.1904	0.1811	0.1609	0.1412	0.1335	0.1492		
	4	0.0443	0.0367	0.0217	0.0112	0.0167	0.0501		
	5	0.1345	0.1291	0.1155	0.0975	0.0789	0.0637		
	6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
10202	1	1.4457	0.5511	0.1492	0.0501	0.0637	0.0000		
	2	0.5511	0.1829	0.0328	0.0116	0.0304	0.0000		
	3	0.1492	0.0328	-0.0049	0.0008	0.0142	0.0000		
	4	0.0501	0.0116	0.0008	0.0042	0.0084	0.0000		
	5	0.0637	0.0304	0.0142	0.0084	0.0059	0.0000		
	6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000		
20101	1	0.7412	0.7463	0.8119	1.0138	1.4275	2.1286		
	2	0.7463	0.7396	0.7789	0.9531	1.3511	2.0618		
	3	0.8119	0.7789	0.7569	0.8616	1.2088	1.9142		
	4	1.0138	0.9531	0.8616	0.8757	1.1313	1.7649		
	5	1.4275	1.3511	1.2088	1.1313	1.2492	1.6931		
	6	2.1286	2.0618	1.9142	1.7649	1.6931	1.7780		
20102	1	2.1286	3.2636	2.4736	0.9033	-0.3028	0.0000		
	2	2.0618	3.2247	2.4521	0.8919	-0.3081	0.0000		
	3	1.9142	3.0847	2.3553	0.8492	-0.3103	0.0000		
	4	1.7649	2.8088	2.1352	0.7630	-0.2884	0.0000		
	5	1.6931	2.3622	1.7434	0.6208	-0.2216	0.0000		
	6	1.7780	1.7098	1.1318	0.4102	-0.0891	0.0000		
20201	1	2.1286	2.0618	1.9142	1.7649	1.6931	1.7780		
	2	3.2636	3.2247	3.0847	2.8088	2.3622	1.7098		
	3	2.4736	2.4521	2.3553	2.1352	1.7434	1.1318		

4	0.9033	0.8919	0.8492	0.7630	0.6208	0.4102
5	-0.3028	-0.3081	-0.3103	-0.2884	-0.2216	-0.0891
6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

20202

1	1.7780	1.7098	1.1318	0.4102	-0.0891	0.0000
2	1.7098	1.0986	0.5999	0.2359	0.0285	0.0000
3	1.1318	0.5999	0.2832	0.1194	0.0458	0.0000
4	0.4102	0.2359	0.1194	0.0490	0.0131	0.0000
5	-0.0891	0.0285	0.0458	0.0131	-0.0190	0.0000
6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

CORE USAGE OBJECT CODE= 105264 BYTES, ARRAY AREA= 160240 BYTES, TOTAL AR
COMPILE TIME= 4.07 SEC, EXECUTION TIME= 8.84 SEC, WATFIV - VERSION 1 L