CRNL--5851

DE82 019853

ORNL-5851

Distribution Category UC-79d

Contract No. W-7405-eng-26

ENGINEERING PHYSICS DIVISION

**AN UPDATED VERSION OF THE DOT 4 ONE- AND TWO-DIMENSIONAL
NEUTRON/PHOTON TRANSPORT CODE**

**W. A. Rhoades
R. L. Childs***

Manuscript Completed: April 1982

Date Published: July 1982

**This Work Sponsored by
DOE Office of Energy Technology,
Division of Reactor Research & Technology,
and The Defense Nuclear Agency**

***Computer Sciences Division.**

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors represented herein do not necessarily state or reflect those of the United States Government or any agency thereof.

**OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37830
operated by
UNION CARBIDE CORPORATION
for the
DEPARTMENT OF ENERGY**

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Reg

TABLE OF CONTENTS

REGION 0	vii
0.1. ABSTRACT	vii
0.2. ACKNOWLEDGEMENTS	vii
REGION 1 - PROGRAM ABSTRACT	1
1.1. Program	1
1.2. Problem Solved	1
1.3. Method of Solution	1
1.4. Related Material	1
1.5. Restrictions	1
1.6. Computers	2
1.7. Running Time	2
1.8. Programming Languages	2
1.9. Operating System	2
1.10. Machine Requirements	2
1.11. Authors	2
1.12. References	3
1.13. Material Available	3
1.14. Abstract Author	3
REGION 2 - INTRODUCTION	4
2.1. The Problem Solved	4
2.2. The Code	4
2.3. The Document	5
REGION 3 - THEORETICAL BASIS	6
3.1. Background	6
3.2. The Integro-Differential Transport Equation	6
3.3. The Finite Difference Formulation	7
3.4. Flux Evaluation Strategy	11
3.5. Interrupted Sweeps	13
3.6. The Linear-Zero Flux Model	13
3.7. The Step Model	19
3.8. Weighted Difference Models	19
3.9. The Source Term	23
3.10. Directional Quadrature Sets	26
3.11. Internal Boundary Sources	27
3.12. External Boundary Conditions	28
3.13. Iteration Strategy	29
3.14. Fission Problems, K-Calculations and Searches	30
3.15. Rebalance and Acceleration	31

3.15.1. Groupwise Rebalance	31
3.15.2. Upscatter Rebalance	32
3.15.3. Space Rebalance	33
3.15.4. Damped Partial Current Rebalance	36
3.15.5. Diffusion Acceleration	37
3.15.6. Source Iteration Acceleration	38
3.16. Negative Source Repair	40
3.17. Spatial Remeshing	40
3.18. Directional Remeshing	41
3.19. Geometric Factors	42
3.20. Diffusion Theory	44
3.21. Units	44
REGION 4 - PROGRAMMER'S INFORMATION	46
4.1. Code Structure	46
4.2. Inter-Machine Adaptability	46
4.3. Service Subroutines	47
4.4. Programming Style and Practices	48
4.5. Overlay Arrangements	49
4.6. Internal Storage Management	49
4.7. Split-Memory Features	52
4.8. External Storage Management	53
4.9. Moment File Management	53
REGION 5 - USER'S INFORMATION	56
5.1. Input Data Specifications	56
5.1.1. Card-Image Format	56
5.1.2. Space Meshes	57
5.1.3. Variable Space Mesh	57
5.1.4. Variable Directional Quadrature	58
5.1.5. Variable Legendre Expansion	58
5.1.6. Adjoin Data	58
5.1.7. Card-Image Input Specifications	58
5.2. Additional Input Data Details	68
5.2.1. Special Geometry Features	68
5.2.2. Special Theory Options	69
5.2.3. Cross Section Input and Mixing	70
5.2.4. Activity Edits	71
5.2.5. Negative Source Removal	72
5.2.6. Problem Type Specification	73
5.2.7. Iteration Limits and Convergence	73
5.2.8. Search Strategy	75
5.2.9. Upscatter Rebalance	76

5.2.10. Fission Rescaling	76
5.2.11. Fission Extrapolation	77
5.2.12. Error-Mode Extrapolation	77
5.2.13. Flux Iteration Acceleration	78
5.2.14. Stabilization	79
5.2.15. Input Source and Flux Specification	79
5.2.16. Key Flux Monitoring	80
5.2.17. Energy Boundary Input	80
5.2.18. Flux Extrapolation Model	81
5.2.19. Alternate Zone Map Input	81
5.3. Details of Job Requirements	82
5.3.1. CPU Time Usage	82
5.3.2. Input and Output Data Files	83
5.3.3. Scratch Data Sets	84
5.3.4. System Buffer Space (NBUF)	84
5.3.5. Memory Requirement - Standard Configurations	85
5.4. Problem Printed Output	87
REGION 6 - OPERATION AT ORNL	93
6.1. Program Access	93
6.2. Disk Space Allocation	95
6.3. I/O Requests and Total Clock Time	95
REFERENCES	99
APPENDIX A. Fido Input	A-1
APPENDIX B. Interface File Specifications	B-1
APPENDIX C. Bibliography of Information Relating to DOT	C-1

Region 0

0.1. Abstract

DOT 4 is designed to allow very large transport problems to be solved on a wide range of computers and memory arrangements. Unusual flexibility in both space-mesh and directional-quadrature specification is allowed. For example, the radial mesh in an R-Z problem can vary with axial position. The directional quadrature can vary with both space and energy group. Several features improve performance on both deep penetration and criticality problems. The program has been checked and used extensively.

0.2. Acknowledgements

Major contributors to previous versions of DOT deserve special recognition. Listed with their last-known location, these include:

W. W. Engle, Jr., ORNL, Oak Ridge, TN
M. L. Gritzner, SAI, LaJolla, CA
F. R. Mynatt, ORNL, Oak Ridge, TN
R. J. Rodgers, United Aircraft Research, Hartford, CT
D. B. Simpson, TVA, Knoxville, TN

Acknowledgement also belongs to E. T. Tomlinson, now with TVA-Chattanooga, who performed an extensive study of the various finite-difference models in this code, and in TWOTRAN II. He contributed the "linear-zero" model as used herein.

Thanks also belong to R. K. Disney, W-ARD, Waltz Mill, for providing the diffusion package from the DOT III code. This package originated with R. Protsik, GE-Sunnyvale. Extensive help in adapting DOT to various computers was received from J. Fletcher, KAPL; M. B. Emmett, ORNL, Oak Ridge; J. T. West, LANL; M. D. Libby, GE-Sunnyvale; D. Matthews, GA-LaJolla; and probably others. Finally, thanks belong to D. R. Vondy of ORNL for many helpful suggestions regarding the acceleration schemes, and to J. K. Lawhorn for the extensive typing and editing task.

Region 1. Program Abstract

1.1. Program: DOT 4

1.2. Problem Solved: DOT 4 determines the flux or fluence of particles throughout a one- or two-dimensional geometric system due to sources either generated as a result of particle interaction with the medium or incident upon the system from extraneous sources. The principal application is to the deep-penetration transport of neutrons and photons. Criticality (k-type and search) problems can also be solved. Numerous printed edits of the results are available, and results can be transferred to output files for subsequent analysis.

1.3. Method of Solution: The Boltzmann transport equation is solved, using either the method of discrete ordinates or the diffusion theory approximation. In the discrete ordinates method, the primary mode of operation, balance equations are solved for the flow of particles moving in a set of discrete directions in each cell of a space mesh, and in each group of a multigroup energy structure. Iterations are performed until all implicitness in the coupling of cells, directions, groups, and source regeneration has been resolved. Several methods are available to accelerate convergence. Anisotropic cross sections can be expressed in a Legendre expansion of arbitrary order. Output data sets can be used to provide an accurate restart of a previous problem or to deliver information to other codes.

Several techniques are available to remove the effects of negative fluxes caused by the finite difference approximation, and of negative scattering sources due to truncation of the cross-section expansion. The space mesh can be described such that the number of first-dimensional (i) intervals varies with the second dimension (j). The number of discrete directions can vary across the space mesh and with energy. Direction sets can be biased, with discrete directions concentrated such as to give fine detail to streaming phenomena.

1.4. Related Material:

Data Files

- Microscopic cross-section input file
- Independent source file (optional)
- Flux guess input file (optional)
- Flux result output files (optional)
- Total source output file (optional)

Related Programs

- DOS - a driver program which coordinates problem execution (IBM only)
- BNDRYS - selects boundary fluxes for subsequent use as internal boundary sources
- GIP - prepares cross-section input from card or tape input
- GR1UNCL - prepares first-collision source due to a point source in RZ geometry (on or off axis)
- RTFLUM - edits flux files and converts between various file formats

1.5. Restrictions: External force fields or nonlinear effects cannot be treated. Flexible dimensioning is used throughout, so that no restrictions are imposed on individual problem parameters. Certain options, especially diffusion theory, are not compatible with variable mesh and quadrature problems.

1.6. Computer: DOT 4 is designed to be applicable to most sophisticated computers which support direct (random) access disk storage or the equivalent. It has special provisions for efficient use of a large, slow memory from which data are moved to fast memory in strings. Certain directly-addressed arrays of low usage can also be kept in slow memory, and slow memory is also used as a buffer between fast memory and certain disk files. Features which may be troublesome on new machines are restricted to replaceable packages. Proper operation has been demonstrated on many types of IBM, CDC, UNIVAC, and CRAY computers.

1.7. Running Time: Running time is roughly proportional to:

Flux work units (FWU) = number of space mesh cells x number of directions x number of energy groups x number of iterations per group.

Depending on the options chosen, a rate of one million FWU per minute on the IBM 370/3033 is typical. Thus, a very large problem with 5,000 space cells, 48 directions, 50 energy groups, 10 iterations per group, and P_3 scattering would require roughly 1 to 1.5 hours of CPU time. Execution on CDC or CRAY computers is usually faster.

1.8. Programming Languages: The standard FORTRAN described in ANS STD.3-1971 is used. The program can be operated with 100% FORTRAN, except for the subroutines necessary to implement I/O. On CRAY, CDC, and UNIVAC systems, these subroutines are obtained from the system library at load time. On IBM systems, they are supplied with the code. On all systems, optional use of non-FORTRAN routines is provided in order to enhance convenience and speed.

1.9. Operating System: No special requirements are made of the operating system. IBM or UNIVAC overlay, CDC overlay, or CDC segmentation facilities may be used if available. The system must support direct (random) access data files.

1.10. Machine Requirements: Memory must be approximately 50,000 words for small problems. The requirement expands with problem size. External data storage must be provided for 9 scratch files, of which 5 must be direct (random) access. User-supplied input and output data files must be supplied on sequential-access devices, i.e., tapes or the equivalent.

1.11. Authors: Many persons have contributed to DOT development since 1965. Major contributors to this version are:

W. A. Rhoades
R. L. Childs

Major contributors to previous versions include:

W. W. Engle, Jr.
M. L. Gritzner
F. R. Myrati
R. J. Rodgers
D. B. Simpson
E. T. Tomlinson

Questions concerning this code should be referred to the Radiation Shielding Information Center (RSIC), Oak Ridge National Laboratory, P. O. Box X, Oak Ridge, Tennessee 37830.

1.12. References:

F. R. Mynatt, F. J. Muckenthaler, and P. N. Stevens, "Development of a Two-Dimensional Discrete Ordinates Transport Theory for Radiation Shielding," CTC-INF-952 (August 1969).

K. D. Lathrop and F. W. Brinkley, "TWOTRAN-II: An Interfaced, Exportable Version of the TWOTRAN Code for Two-Dimensional Transport," LA-4848-MS (July 1973).

W. A. Rhoades and R. L. Childs, "An Updated Version of the DOT 4 One- and Two-Dimensional Neutron/Photon Transport Code," ORNL-5851 (to be published).

1.13. Material Available:

User's Manual
Source File in Update Form
Job Control Language for Unloading Source File and Installing Code
Sample Problem Solutions

1.14. Abstract Author: W. A. Rhoades, ORNL.

Region 2. Introduction

2.1. The Problem Solved

DOT 4 is primarily intended to solve large neutron and photon transport problems on a wide variety of computer types using the method of discrete ordinates. Extensive provisions are made to facilitate the specification of these large problems and to accommodate the problems to the given computer system. Although most DOT 4 problems are ex-core transport problems, the code also has K_{eff} capability and various types of searches. A diffusion-theory section is valuable for initiating eigenvalue problems or as an alternative to the primary discrete ordinates section when applicable.

Special remeshing features allow the number of first-dimension (i) mesh intervals to vary with the second dimension (j) index. The directional quadrature set can be chosen from an arbitrary number of input sets. The choice can vary with spatial position and with energy group. This flexibility has proven quite effective in concentrating CPU effort in areas needing attention, such as streaming gaps. Unsymmetric direction sets can be used when streaming is primarily either upward or downward.

A variety of options allow sources to be specified at internal or external boundaries, distributed by space and energy, or determined from an input flux guess. A similar variety of output source information to be used in coupling to other problems is provided. Typical applications include "bootstrapping," in which a very large problem is solved in several segments; core-vessel combinations, in which a core is solved in an eigenvalue calculation and then represented by internal boundary conditions in a fixed-source vessel calculation; and coupled discrete-ordinates-Monte Carlo calculations.

"First-collision-source" data, actually an analytical first-flight scattering source, can be accommodated. Output files for an "analytical-last-flight" integration can be obtained. These features provide increased accuracy when dealing with out-of-system or localized sources or detector locations.

A single output file contains both distributed flux moments and boundary directional fluxes, so that this file, plus the original input data, provide an "exact" restart. This allows a few groups of a large problem to be solved in each computer run without loss of efficiency.

Both 1-D and 2-D geometries can be treated. The diffusion-theory section has triangular-mesh capability as well. A powerful slab reflection/transmission feature is available. A variety of acceleration options are available. Extensive use of input options and output edits give the user very direct control over the iteration process. While this places a burden of decision-making on him, it is essential to the solution of large and difficult problems. Default and recommended values assist the uninitiated in solving less heroic problems.

2.2. The Code

The code has been adapted to a large variety of computers. The "unified source file" concept involves maintaining only a single version of the code which is configured to a specific computer at distribution time. Standard language and programming practices are

used throughout. In-stream "language flags" make minor language adjustments as required when a specific configuration is tailored.

The standard code can be operated with only input-output (I/O) routines supplied as system-dependent. In the usual case, a small number of additional system-dependent routines provide increased speed and convenience. CDC configurations can be tailored to make efficient use of the slow memory on the CDC 7600. On all configurations, direct (random) access scratch files are used as an extension of memory. Large buffers controlled by the program routines provide efficient use of these files.

The code attempts to make the maximum use of available memory in order to minimize I/O costs and delays. Reversion to more extensive use of scratch-files is automatic when required.

The flexibility provided, together with the close control of the solution process, requires an extensive and complicated input data list. Most of these data can be ignored for most problems. A diverse set of sample problems serves as a guide to actual requirements for practical problems.

An earlier version of DOT 4 was reported in Reference 1.

2.3. The Document

The remainder of this document is divided into theoretical, programmer's and user's information sections, followed with a section describing local application procedures at ORNL. The user's section is intended to be very thorough, containing details, such as the definition of the various convergence parameters, which might have been given in the theoretical section. The intent is to make the user's section sufficient to guide correct use of the code. The theoretical section passes lightly over derivations which are covered in detail in references and dwells on features which are peculiar to DOT 4. The programmer's section is more brief than some since much of the descriptive material which it might include is contained in or distributed with the source program. Such material is more easily kept current and correct in that form.

Portions of the document may seem wordy or trivial to the expert, but it is not to that audience that the document is aimed. It is intended to guide the uninitiated user in understanding the function and proper use of this code.

Region 3. Theoretical Basis

3.1. Background

A summary of the mathematical basis for the method of discrete ordinates as we use it today is given by Lathrop and Brinkley.² Mynatt et al.³ give a full mathematical development, applications information, and comparisons of results with experiments. Carlson and Lathrop⁴ give a full development of the method with emphasis on physical principles and evaluation techniques. They also give an annotated bibliography of early work in the field including a very early report by Carlson in 1953.⁵ A 1958 paper by Carlson and Bell⁶ may have been the first widely-distributed description of the "discrete S_n method" as they called it. In 1960, Carlson, Lee, and Worlton published a description of the DSN code, which used their method.⁷ Davison,⁸ in 1957, gave a thorough review of fundamental discrete-ordinates theory, and a review of Carlson's work in an Appendix. Davison acknowledges a 1953 book by Chandrasekhar⁹ as the origin of the method as he presents it, and a 1943 paper by Wick¹⁰ as the origin of the basic idea.

In this report, we follow the notation of Lathrop and Brinkley to some extent. Their TWOTRAN II code was useful to us in constructing DOT 4, and it may be worthwhile to comment on the relationship between TWOTRAN and DOT. The earlier versions of DOT actually descended from the LASL codes DTK and DDK, and from the LASL/UNC code DDF/2DF. A version of DOT 4 was distributed informally in 1976, although formal publication was not made until the DOT III version in 1973.¹¹ Early versions of TWOTRAN were published in a 1968 GA report¹² and elsewhere.

Descendants of both TWOTRAN and DOT remain in widespread use today, and their very different capabilities apparently justify this dual existence. Numerous other writings pertinent to DOT development appear in ANS Transactions and elsewhere. A reasonably complete bibliography of related material is given in Appendix C.

3.2. The Integro-Differential Transport Equation

The Boltzmann transport equation, as applied to static neutron and photon transport,² can be expressed:

$$\begin{aligned} \nabla \cdot \underline{\Omega} \psi(\underline{r}, E, \underline{\Omega}) + \sigma^T(\underline{r}, E, \underline{\Omega}) \psi(\underline{r}, E, \underline{\Omega}) = \\ \int \int \sigma^s(\underline{r}, E, E', \underline{\Omega}, \underline{\Omega}') \psi(\underline{r}, E', \underline{\Omega}') d\underline{\Omega}' dE' \\ + \frac{1}{4\pi} \chi(\underline{r}, E) \int \int \nu(\underline{r}, E') \sigma^F(\underline{r}, E') \psi(\underline{r}, E', \underline{\Omega}') d\underline{\Omega}' dE' + Q(\underline{r}, E, \underline{\Omega}) \end{aligned} \quad (3.1)$$

In this, $d\underline{\Omega}$ is a differential solid angle about the direction vector $\underline{\Omega}$, dE is a differential energy about E , and $\underline{r}$ is a position vector with reference to an arbitrary origin. The directional flux function, ψ , is defined such that the number of particles moving in volume

$d\mathbf{r}$ about $\mathbf{r}$ with energies falling within dE and directions of motion within $d\Omega$ will be $\psi d\Omega dE d\mathbf{r}$. Also, σ^T and σ^F are macroscopic cross sections for total interaction and fission, respectively, and σ^s is the cross section for scattering from energy E' and direction Ω' to energy E and direction Ω . The functions ν and χ represent the total fission yield of secondary particles and the corresponding energy distribution. Q represents an extraneous source, if any.

Although the general Boltzmann equation includes time dependence, removal of that dependence in the static form presented here is an important simplification. Certain other simplifications will also be made later. As a practical matter, scattering is generally independent of initial direction, so that it can be expressed in terms of the single scattering angle:

$$\mu_0 = \Omega' \cdot \Omega \quad (3.2)$$

rather than as a function of two direction vectors. Although some applications exist for the dependence of σ^s on Ω , they are unusual, and this dependence is not allowed in DOT 4. A somewhat weaker justification exists for the removal of the dependence of χ on space. In fact, if more than one fissile species exists in the system, and if the species do not always fission in the same proportion, χ will depend on space. Nonetheless, the code does not presently allow such dependence.

3.3. The Finite Difference Formulation

The procedures for obtaining a finite-difference equation from Eq. 3.1 are discussed thoroughly in References 3 and 4, although the form presented here follows Reference 2:

$$\begin{aligned} W_m \mu_m \left\{ A_{i+\frac{1}{2}j} N_{i+\frac{1}{2}j,m,g} - A_{i-\frac{1}{2}j} N_{i-\frac{1}{2}j,m,g} \right\} \\ + W_m \tau_m \left\{ B_{i,j+\frac{1}{2}} N_{i,j+\frac{1}{2},m,g} - B_{i,j-\frac{1}{2}} N_{i,j-\frac{1}{2},m,g} \right\} \\ + \left\{ A_{i+\frac{1}{2}j} - A_{i-\frac{1}{2}j} \right\} \left\{ \alpha_{m+\frac{1}{2}} N_{i,j,m+\frac{1}{2},g} - \alpha_{m-\frac{1}{2}} N_{i,j,m-\frac{1}{2},g} \right\} \\ + \sigma_{i,j,g}^T W_m V_{i,j} N_{i,j,m,g} = W_m V_{i,j} S_{i,j,m,g} \end{aligned} \quad (3.3)$$

It can be shown that the formulation of Mynatt et al.³ is equivalent to this.

In this notation, subscripts i and j represent mesh intervals in the first and second space dimensions. The subscript m refers to the direction in a ordered set of directions along which flux is to be evaluated. Subscripts such as $i + 1/2$ refer to the interval boundaries, e.g. boundary $i + 1/2$ is the boundary separating interval i from interval $i + 1$. In the case of m , this is somewhat artificial, since each m represents a discrete direction with no defined sector of solid angle associated. In fact, directions represented by

successive m 's may not even be "adjacent" in direction space. Even so, the equation requires coupling between certain fluxes having consecutive m values, and the coupling terms, α , will be defined in a consistent manner. Subscripts such as $m + 1/2$, then, merely denote artificial intermediate values. The subscript g refers to energy group.

It may be noted at this point that obvious subscripts are frequently omitted in discrete-ordinates writings, e.g. the first term of 3.3 might later be written:

$$W_{ij} \left(A_{i+1/2} N_{i+1/2} - A_{i-1/2} N_{i-1/2} \right)$$

In practice, this approach is probably less confusing than the inclusion of long lists of redundant subscripts.

In certain discussions, I and J are understood to be the upper limits of i and j . It may also be noted that space mesh positions are discussed such that boundaries having $i=1/2$, $i=I+1/2$, $j=1/2$, and $j=J+1/2$ are spoken of as "left," "right," "bottom," and "top," boundaries, regardless of geometry.

The ordered set of directions of particle travel are characterized by their direction cosines, (μ_m, ξ_m, η_m) . In all geometries μ_m is the cosine of the angle which the direction of travel makes with first-dimension axis, i.e., X or R , while η is the cosine of the angle with the Z axis (Table 3.1). In each case, ξ is the cosine with the remaining direction vector, either along the Y axis, or in the azimuthal direction in the case of $R\theta$.

Due to symmetry, no flow is associated with one of the direction cosines, either η or ξ . Thus, τ is used to represent either ξ or η , as shown in Table 3.1. Also, ω is allowed to represent the remaining cosine. The parameters A and B are cell areas perpendicular to the axes from which μ and τ are measured.

Each direction has an associated weight, W_m . Quantities involving integrals over all directions are to be evaluated by sums with W_m as the weighting function. N represents the directional flux, ψ , in direction, m , although it will be seen that N has different units. V and S are the volume and source in a given mesh cell, and σ^T is the macroscopic total cross section. It is assumed that S includes extraneous sources, scattering from other energy groups, scattering at the given energy from other directions, fission, and any other particle sources, e.g. $(n,2n)$.

As indicated in Table 3.1, the 1-D geometries evaluate flux dependence in only the first dimension. In such cases, the usual evaluation procedures are used, but $B=0$.

It can be observed that, except for the terms in α , Eq. 3.1 is simply a statement of particle balance for a given cell, and could have been written down directly. From this, it is evident that the discrete-ordinates equation conserves particles inherently, if the α terms are defined such as to conserve.

The terms in α are effective only in curved geometry, RZ or $R\theta$. Their physical meaning can be made apparent by considering a particle moving inward along a path which will take it closer to the axis of a cylinder. While η is a constant for such a path, μ constantly increases from negative, through zero, and ultimately toward $\mu = +1$ as the particle moves away from the centerline. Thus, the discrete-ordinates equation must show a comparable flow toward directions of larger μ for comparable particle paths.

Table 3.1. Directional Properties of the Geometries

	1-D			2-D		
Geometry	XZ	RZ	R θ	XZ	RZ	R θ
Dimensions Solved	X	R	R	X,Z	R,Z	R, θ
Dimension Corresponding to μ	X	R	R	X	R	R
Dimension Corresponding to ξ	Y	θ	θ	Y	θ	θ
Dimension Corresponding to η	Z	Z	Z	Z	Z	Z
Cosine Corresponding to τ	η	η	ξ	η	η	ξ
Cosine Corresponding to ω	ξ	ξ	η	ξ	ξ	η
Dimension Perpendicular to A	X	R	R	X	R	R
Dimension Perpendicular to B	Z	Z	θ	Z	Z	θ

In order to allow manageable evaluation of the terms in α , an ordering discipline is imposed upon the directions represented by m . Since the α terms must couple directions of like η , directions of like η are grouped contiguously, in order of increasing μ . The groups of directions with like η (" η levels") are ordered such that all directions with $\eta < 0$ precede all directions with $\eta > 0$. Thus, the α -flow, following an actual path inward and then outward, is always toward directions of increasing m within an η level. It can also be seen that there must be no flow between η levels in order to be consistent with the physical model.

The recursion relationship for the α 's can be obtained at this point by observing that Eq. 3.3 must be valid in a source-free region where N is uniform. Since no constraint may be placed upon σ^T , A , B , or V , then it must follow that, for a given η level:

$$-W_m \mu_m = \alpha_{m+1/2} - \alpha_{m-1/2} \quad (3.4)$$

Since it is required that α coupling be only between directions within an η level, then $\alpha_{m-1/2} = 0$ for the first direction of the level. It is essential that, for each level:

$$\sum_m W_m \mu_m = 0 \quad (3.5)$$

so that $\alpha_{m+1/2}$ will be 0 for the last m of that level, and there will be no net flow into or out of the η level. This insures that particle conservation is rigorously met. Symmetrical quadrature sets always obey this condition, and it is important that unsymmetrical sets obey this condition, also.

There is certain advantage to be gained by substituting a new variable, β for α :

$$\beta_m = \frac{1}{W_m} \left(\alpha_{m+1/2} + \alpha_{m-1/2} \right) . \quad (3.6)$$

The recursion relationship for the β_m can be obtained by applying its definition to the recursion relationship for α :

$$W_m \beta_m = \alpha_{m+1/2} + \alpha_{m-1/2} , \quad (3.7)$$

$$W_{m+1} \beta_{m+1} = \alpha_{m+3/2} + \alpha_{m+1/2} , \quad (3.8)$$

$$\alpha_{m+1/2} = \alpha_{m-1/2} - W_m \mu_m , \quad (3.9)$$

and

$$\alpha_{m+3/2} = \alpha_{m+1/2} - W_{m+1} \mu_{m+1} . \quad (3.10)$$

Therefore

$$W_{m+1} \beta_{m+1} = W_m \beta_m - \left(W_{m+1} \mu_{m+1} + W_m \mu_m \right) . \quad (3.11)$$

Recalling that $\alpha_{m-1/2} = 0$ for the first direction of each η level, then it follows that $\beta_m = -\mu_m$ for such levels; at least if $W_m \neq 0$. As will be discussed later, this relationship is assumed even when $W_m = 0$.

It is important, both in applying boundary conditions and in obtaining proper stability, to "follow the flow" in writing the recursion formulas; i.e., all directions with $\mu < 0$ must be evaluated from large X or K values toward small, and so with the other dimension. Similarly, positive μ values require evaluation from small X to large, etc. These procedures can be written in a single formula by use of subscript increments defined in terms of the signum function (sg); which has value ± 1 according to the sign of its argument:

$$c = \frac{1}{2} \text{Sg}(\mu_m) ; \quad d = \frac{1}{2} \text{Sg}(\tau_m) . \quad (3.12)$$

Written in terms of β , c , and d , Eq. 3.3 can be simplified to:

$$\begin{aligned} & |\mu| \left[A_{i+c} N_{i+c} - A_{i-c} N_{i-c} \right] + |\tau| B \left[N_{j+d} - N_{j-d} \right] \\ & + \frac{1}{2} \Delta A \left[\left(\beta_m - \mu_m \right) N_{m+1/2} - \left(\beta_m + \mu_m \right) N_{m-1/2} \right] + V \sigma^T N = VS \end{aligned} \quad (3.13)$$

where

$$\Delta A = A_{i+1/2} - A_{i-1/2} \quad (2.14)$$

and where N is simply a simplified representation of $N_{i,j,m}$, etc.

3.4. Flux Evaluation Strategy

Evaluation "following the flow," as discussed above, is also evaluation "sweeping away from boundary conditions into the mesh." Since evaluation always begins with $m = 1$, the first sweep must be toward smaller values of i and j . Where I and J represent the largest values of i and j , then the boundary values $N_{I+1/2}$ and $N_{J+1/2}$ are decided *a priori* based on physical considerations at the outer boundaries of the system. No such physical considerations govern the boundary flux at $m = 1/2$, however, as will be seen.

Before dealing with $N_{1/2}$, let us observe that, even with N_{I-c} and N_{J-d} known from boundary considerations and $N_{m-1/2}$ to be defined by a yet-unspecified process, Eq. 3.3 would still involve 4 unknowns. The three advance boundary values, N_{I+c} , N_{J+d} , $N_{m+1/2}$ have been treated as completely independent, and are, therefore, not completely defined by Eq. 3.3. In fact, all of the values of N represent values of a single continuous function ψ at adjacent locations, and cannot vary more freely than ψ does. If we suspect that ψ can be approximated adequately by straight-line segments between adjacent boundaries, then the following "linear" or "diamond" difference model results:

$$N = \frac{1}{2} \left(N_{I+c} - N_{I-c} \right) , \quad (3.15a)$$

$$N = \frac{1}{2} \left(N_{J+d} - N_{J-d} \right) , \quad (3.15b)$$

and

$$N = \frac{1}{2} \left(N_{m+1/2} - N_{m-1/2} \right) , \quad (3.15c)$$

As previously discussed, $\beta_1 = -\mu_1$. Accordingly, Eqs. 3.13 and 3.15 cannot yield direct information as to an appropriate value of $N_{1/2}$, and the other results are independent of this value. Accordingly, $N_{1/2}$ is arbitrary and unknown. If N_1 is to be related to $N_{1/2}$ and $N_{3/2}$, the only acceptable assumption is:

$$N_{3/2} = N_1. \quad (3.16)$$

Since each η level begins with a $\beta_m = -\mu_m$, each must start with an "initiating direction" having:

$$N_{m+1/2} = N_m ; \quad \beta = -\mu . \quad (3.17)$$

If an initiating direction occurs for $m > 1$, and m' is the predecessor to m , then Eq. 3.15 defines values for $m' + 1/2$ and $m - 1/2$ which do not agree. In fact, the flux is not required to be continuous between directions on different η levels, and they do not, in general, describe adjacent directions. Thus, there is neither physical nor mathematical requirement for continuity of flux between them.

It has been traditional in the early DOT codes to allow these initiating directions to have $W_m = 0$, i.e., zero-weight directions, and to adjust their directions such that $\xi = 0$. Carlson and Lathrop describe this procedure.⁴ On the other hand, TWOTRAN II uses directions having small, finite weights and having W_m small, but not 0. Tomlinson et al.¹³ showed that, other factors being equal, this choice does not affect the results of test problems significantly. Both types of calculations are available in DOT 4, although the direction sets input to the code must always contain directions with $W_m = 0$ to signal the beginning of a new η level. When TWOTRAN-like calculations are done, or in XZ geometry, flux is not calculated in directions having $W_m = 0$.

Having decided on boundary values for initiating the sweeps, Eqs. 3.15 and 3.17 define the remaining unknowns in terms of the single unknown, N , for the initial direction of each η level:

$$N_{i+c} = 2N - N_{i-c} \quad , \quad (3.18)$$

$$N_{j+d} = 2N - N_{j-d} \quad . \quad (3.19)$$

Defining, as a matter of convenience:

$$\bar{A} = \frac{1}{2}(A_{i+1/2} + A_{i-1/2}) \quad , \quad (3.20)$$

observing that:

$$2|\mu|A_{i+c} + \Delta A(\beta - \mu) = 2|\mu|\bar{A} + \Delta A\beta \quad , \quad (3.21)$$

and recalling that $\beta = -\mu$ for the initial direction of each η -level, the value of N can be written explicitly:

$$N = \frac{VS + 2|\mu|\bar{A}N_{i-c} + 2|\tau|BN_{j-d}}{V\sigma^T + 2|\mu|\bar{A} + 2|\tau|B} \quad ; \quad \beta = -\mu \quad . \quad (3.22)$$

This equation applies to the initiating directions, whether $W = 0$ for such directions or not. For all other directions, Eq. 3.17 does not apply, and $N_{m-1/2}$ is obtained by applying Eq. 3.15c to the preceding directions:

$$N_{m+1/2} = 2N - N_{m-1/2} \quad ; \quad \beta \neq -\mu \quad . \quad (3.23)$$

Since $\beta \neq -\mu$ for such directions, terms in β appear in the formulation for N :

$$N = \frac{VS + 2|\mu|AN_{i-c} + 2|\tau|BN_{j-d} + \Delta A\beta N_{m-1/2}}{V\tau^T + 2|\mu|A + 2|\tau|B + \Delta A\beta} ; \beta \neq -\mu \quad (3.24)$$

At this point, an explicit sweep pattern has been found. Determining $N_{m=1/2}$, $N_{I+1/2}$, and $N_{J+1/2}$ as described, Eq. 3.22 gives $N_{I,J,1}$ explicitly. Equation 3.18 gives $N_{I-1/2}$ explicitly, recalling that $\mu < 0$ for the first direction, and thus $c = -1/2$. This evaluation can proceed for all i 's down to the boundary value at $i = 1/2$. With these values in storage, values for $m = 1 1/2$ can be found using Eq. 3.17. If that η -level contains additional directions with $\mu < 0$, values for $m = 2$ are evaluated from Eq. 3.24, and extrapolated with Eq. 3.23. This process continues for all m 's for which $\eta < 0$ and $\mu < 0$. Having these values, the values for directions having $\eta < 0$ and $\mu > 0$ at $i = 1/2$ can be determined from physical boundary conditions. The sweep proceeds as before, except, of course, that evaluation is from $i = 1/2$ to $i = I + 1/2$. From these values, Eq. 3.19 determines $N_{J-1/2}$, and the process proceeds as before downward to $j = 1/2$. Boundary values at $j = 1/2$ for directions having $\eta > 0$ are determined from physical boundary conditions at the bottom boundary, and the sweep resumes, this time from $j = 1/2$ to $j = J + 1/2$.

3.5. Interrupted Sweeps

The sweep process described above assumes that the same direction set would be used continuously throughout the space mesh, but this is not necessary. If more than one direction set is used along a row of the mesh, the sweep can be interrupted to perform "remeshing" operations, i.e., to transform $N_{m+1/2}$ from one direction set to another. If we call a set of contiguous i values having a common direction set an I block, the general mesh sweep ordering is as shown in Fig. 3.1. The remeshing operations are discussed in a later section.

A similar remeshing can be made between j levels simply by transforming the flux at $j + c$ from one direction set to another before it is used in the recursion for the next row. DOT 4 allows both types of transformations. In addition, it allows the use of i -meshes which do not match at the boundary $j + d$. It is apparent that the sweep algorithm allows this. After calculating row j , fluxes for $j + d$ are determined from Eq. 3.19. If j' is the successor to j , a new set of fluxes in the new direction set and i -mesh appropriate to j' are defined at $j' - d$ such as to preserve important physical properties of the flux at $j + d$. These new fluxes are used in Eq. 3.22 or 3.24 as appropriate, and the sweep proceeds as before.

3.6. The Linear-Zero Flux Model

It is clear from physical considerations that, given positive sources, boundary values, and cross sections, we expect N to be everywhere positive. Certainly Eqs. 3.22 and 3.24 will produce positive N 's under such conditions. Unfortunately the "extrapolation equations," 3.18, 3.19, and 3.23 will not always do so. If:

$$N < \frac{1}{2} \max \left(N_{i-c}, N_{j-d}, N_{m-1/2} \right) \quad (3.25)$$

Fig. 3.1. Mesh Sweep Ordering.

then at least one value produced by the extrapolations will be negative. This may result in negative values of N in successive applications of Eq. 3.22 or 3.24. These false values relate to the assumption, made in writing Eq. 3.15, that flux could be adequately represented by a straight-line segment between boundaries. If the total cross section becomes sufficiently large, for example, that assumption is no longer valid. In Eq. 3.22, the value of N can be made arbitrarily close to 0 by a large value of σ^T , as is well known. Also, if S and N_{j-d} are 0, but B is not 0, negative values of N_{i+c} will result for all directions having

$$|\mu|A < |\tau|B \quad (3.26)$$

no matter what non-negative value of σ^T is used. Thus, negatives can occur in a void. If false negatives are so easily generated, it is apparent that inaccurate positive numbers can also be generated, and these errors are not easily detected. These considerations make Eqs. 3.22 and 3.24 unsuitable for use as given. More complexity is the only remedy.

The TWOTRAN code pioneered an approach called the "zero-fixup" or "linear-zero" model. As defined by Tomlinson et al.,¹³ this method requires that the values of N_{i+c} , N_{j+d} , and $N_{m+1/2}$ be calculated as described in the "linear" model above. If any boundary flux is negative, it is set to zero, and N is recalculated with that assumption. If the new N is negative, the fix-up attempt is abandoned. This can happen only if negative sources are present, and such a condition is considered unrepairable.

If N is non-negative, the remaining two boundary fluxes are extrapolated using the linear models. If either of these new fluxes is negative, N is recalculated assuming two fluxes are zero. If the new N is negative, the fixup attempt is abandoned. If not, the remaining flux is extrapolated. If it is negative, then all three boundary fluxes are set to zero, and N is recomputed (assuming the total cross section is not zero). If this recomputed N is zero, the fixup is abandoned.

Seven sets of equations result from the zero fix-up strategy, and special equations are required to evaluate N for boundary directions, i.e., if $\beta = -\mu$:

(1) If $N_{i+c} < 0$, use:

$$N_{i+c} = 0.0 \quad (3.26a)$$

$$N_{j+d} = 2N - N_{j-d} \quad (3.26b)$$

and, if $\beta \neq \mu$:

$$N_{m+1/2} = 2N - N_{m-1/2} \quad (3.26c)$$

and

$$N = \frac{VS + |\mu|A_{i-c}N_{i-c} + 2|\tau|BN_{j-d} + \Delta A\beta N_{m-1/2}}{V\sigma^T + 2|\tau|B + \Delta A(\beta - \mu)} \quad (3.26d)$$

or, if $\beta = -\mu$:

$$N_{m+1/2} = N \quad (3.26e)$$

and

$$N = \frac{VS + |\mu|A_{i-c}N_{i-c} + 2|\tau|B^*V_{j-d}}{V\sigma^T + 2|\tau|B - \Delta A} \quad (3.26f)$$

(2) If $N_{j+d} < 0$, use:

$$N_{i+c} = 2N - N_{i-c} \quad , \quad (3.27a)$$

$$N_{j+d} = 0.0 \quad , \quad (3.27b)$$

and, if $\beta \neq -\mu$:

$$N_{m+1/2} = 2N - N_{m-1/2} \quad (3.27c)$$

and

$$N = \frac{VS + 2|\mu|AN_{i-c} + |\tau|BN_{j-d} + \Delta A\beta^*V_{m-1/2}}{V\sigma^T + 2|\mu|A + \Delta A\beta} \quad ; \quad (3.27d)$$

or, if $\beta = -\mu$:

$$N_{m+1/2} = N \quad (3.27e)$$

and

$$N = \frac{VS + 2|\mu|AN_{i-c} + |\tau|BN_{j-d}}{V\sigma^T + 2|\mu|A} \quad (3.27f)$$

(3) If $N_{m+1/2} < 0$, use:

$$N_{i+c} = 2N - N_{i-c} \quad , \quad (3.28a)$$

$$N_{j+d} = 2N - N_{j-d} \quad , \quad (3.28b)$$

$$N_{m+1/2} = 0.0 \quad , \quad (3.28c)$$

and, if $\beta \neq -\mu$:

$$N = \frac{VS + 2|\mu|AN_{i-c} + 2|\tau|BN_{j-d} + \frac{1}{2}\Delta A(\beta + \mu)N_{m-\frac{1}{2}}}{V\sigma^T + 2|\mu|\bar{A} + 2|\tau|B + \Delta A\mu} ; \quad (3.28d)$$

or, if $\beta = -\mu$:

$$N = \frac{VS + 2|\mu|AN_{i-c} + 2|\tau|BN_{j-d}}{V\sigma^T + 2|\mu|\bar{A} + 2|\tau|B} \quad (3.28e)$$

(4) If $N_{i+c} < 0$ and $N_{j+d} < 0$:

$$N_{i+c} = 0.0 , \quad (3.29a)$$

$$N_{j+d} = 0.0 , \quad (3.29b)$$

and, if $\beta \neq -\mu$:

$$N_{m+\frac{1}{2}} = 2N - N_{m-\frac{1}{2}} \quad (3.29c)$$

and

$$N = \frac{VS + |\mu|A_{i-c}N_{i-c} + |\tau|BN_{j-d} + \Delta A\beta N_{m-\frac{1}{2}}}{V\sigma^T + \Delta A(\beta - \mu)} ; \quad (3.29d)$$

or, if $\beta = -\mu$:

$$N_{m+\frac{1}{2}} = N , \quad (3.29e)$$

and

$$N = \frac{VS + |\mu|A_{i-c}N_{i-c} + |\tau|BN_{j-d}}{V\sigma^T - \Delta A\mu} \quad (3.29f)$$

(5) If $N_{i+c} < 0$ and $N_{m+\frac{1}{2}} < 0$, use:

$$N_{i+c} = 0.0 , \quad (3.30a)$$

$$N_{j+d} = 2N - N_{j-d} , \quad (3.30b)$$

$$N_{m+\frac{1}{2}} = 0.0 , \quad (3.30c)$$

and, if $\beta \neq \mu$:

$$N = \frac{VS + |\mu|A_{i-c}N_{i-c} + 2|\tau|BN_{j-d} + \frac{1}{2}\Delta A(\beta+\mu)N_{m-\frac{1}{2}}}{V\sigma^T + 2|\tau|B} ; \quad (3.30d)$$

or, if $\beta = \mu$, use Eq. 3.26f.

(6) If $N_{j+d} < 0$ and $N_{m+\frac{1}{2}} < 0$, use:

$$N_{i+c} = 2N - N_{i-c} , \quad (3.31a)$$

$$N_{j+d} = 0.0 , \quad (3.31b)$$

$$N_{m+\frac{1}{2}} = 0.0 , \quad (3.31c)$$

and, if $\beta \neq \mu$:

$$N = \frac{VS + 2|\mu|AN_{i-c} + |\tau|BN_{j-d} + \frac{1}{2}\Delta A(\beta+\mu)N_{m-\frac{1}{2}}}{V\sigma^T + 2|\mu|A + \Delta A\mu} ; \quad (3.31d)$$

or, if $\beta = \mu$, use Eq. 3.27f.

(7) If $N_{i+c} = N_{j+c} = N_{m+\frac{1}{2}} = 0$, use

$$N_{i+c} = 0.0 , \quad (3.32a)$$

$$N_{j+d} = 0.0 , \quad (3.32b)$$

$$N_{m+\frac{1}{2}} = 0.0 , \quad (3.32c)$$

and, if $\beta \neq -\mu$:

$$N = \frac{VS + |\mu|A_{i-c}N_{i-c} + |\tau|BN_{j-d} + \frac{1}{2}\Delta A(\beta+\mu)N_{m-\frac{1}{2}}}{V\sigma^T} ; \quad (3.32d)$$

or, if $\beta = \mu$, use Eq. 3.29f.

The linear-zero model always produces non-negative results at the extrapolated boundaries, given non-negative sources, and a great many successful calculations are done with a similar model used in TWOTRAN. Unfortunately, it is not satisfactory for most of the calculations performed with DOT. In practical calculations, the source term, S , involves sums

over other energy groups and directions. The mesh sweep must be solved iteratively until it converges, i.e., until the fluxes calculated from S agree with the fluxes from which S was calculated. In the difficult problems for which DOT is designed, it is not unusual for values of a given flux to switch back and forth between 0 and a finite value in successive iterations. Several fluxes may "resonate" with each other, the switching in one exciting the opposite switching in the other, and convergence can be thwarted indefinitely. Because of this, the usefulness of the linear-zero model is limited generally to reactor core calculations, for which it is usually fast and accurate.

3.7. The Step Model

The "step" model is much simpler. If the approach of Eq. 3.23 is arbitrarily extended to the space dimensions:

$$N_{i+c} = N \quad (3.33a)$$

$$N_{j+d} = N \quad (3.33b)$$

$$N_{m+\frac{1}{2}} = N \quad (3.33c)$$

Then Eqs. 3.24 and 3.22 can be solved explicitly for N .

If $\beta \neq -\mu$:

$$N = \frac{VS + |\mu|A_{i-c}N_{i-c} + |\tau|BN_{j-d} + \frac{1}{2}\Delta A(\beta + \mu)N_{m+\frac{1}{2}}}{V\sigma^T + |\mu|A_{i-c} + |\tau|B + \frac{1}{2}\Delta A(\beta + \mu)} \quad (3.34a)$$

and if $\beta = -\mu$:

$$N = \frac{VS + |\mu|A_{i-c}N_{i-c} + |\tau|BN_{j-d}}{V\sigma^T + |\mu|A_{i-c} + |\tau|B} \quad (3.34b)$$

Equation 3.33 gives the new values of N_{i+c} , etc., and the calculation proceeds. The "switching-oscillation" common to the linear-zero model is impossible, and the results are always positive, given non-negative sources. Unfortunately, this model requires such small mesh intervals for practical calculations that it is almost never used.

3.8. Weighted Difference Models

Most serious DOT calculations are performed with some form of "weighted-difference" model. As will be seen, adjustable parameters are used in the extrapolation equations such as to provide non-negative fluxes. This does not quite assure that the non-negative fluxes will be accurate, but several studies indicate that it is the most satisfactory of the DOT models.^{14,15,16} As will be seen, two slightly different weighted models are available.

To derive the weighted-difference models:

define, where $\beta \neq -\mu$:

$$a = \frac{N - N_{i-c}}{N_{i+c} - N_{i-c}}, \quad (3.35a)$$

$$b = \frac{N - N_{j-d}}{N_{j+d} - N_{j-d}}, \quad (3.35b)$$

and

$$c = \frac{N - N_{m-\frac{1}{2}}}{N_{m+\frac{1}{2}} - N_{m-\frac{1}{2}}}, \quad (3.35c)$$

from which it follows:

$$N_{i+c} = \frac{1}{a} N - \left(\frac{1-a}{a} \right) N_{i-c}, \quad (3.36a)$$

$$N_{j+d} = \frac{1}{b} N - \left(\frac{1-b}{b} \right) N_{j-d}, \quad (3.36b)$$

and

$$N_{m+\frac{1}{2}} = \frac{1}{c} N - \left(\frac{1-c}{c} \right) N_{m-\frac{1}{2}}. \quad (3.36c)$$

With these substitutions into Eq. 3.13:

$$N = \frac{VS + X'N_{i-c} + Y'N_{j-d} + Z'N_{m-\frac{1}{2}}}{V\sigma^2 + X' + Y' + Z'} \quad (3.37)$$

where

$$X' = |\mu| \left[(1/a - 1) A_{i+c} + A_{i-c} \right] \quad (3.38a)$$

$$Y' = |\tau| \left[(1/b - 1) B_{j+d} + B_{j-d} \right] \quad (3.38b)$$

$$Z' = \frac{1}{2} \Delta A \left[(1/c - 1)(\beta - \mu) + (\beta + \mu) \right] \quad (3.38c)$$

The values of a , b , and c are determined by substituting Eq. 3.36 into Eq. 3.37, and then, by examination, choosing these coefficients to ensure non-negativity of the boundary flux. Sufficient conditions are:

$$1-a = \frac{SV\theta_s + \left[|r|BN_{j-d} + \frac{1}{2}\Delta A(\beta + \mu) N_{m-\frac{1}{2}} \right] \theta_n + |\mu|A_{i-c}N_{i-c}}{2 \left[\frac{1}{2}V\sigma^T + |r|B + \frac{1}{2}\Delta A(\beta - \mu) \right] N_{i-c}} \quad (3.39a)$$

$$1-b = \frac{SV\theta_s + \left[|\mu|A_{i-c}N_{i-c} + \frac{1}{2}\Delta A(\beta + \mu) N_{m-\frac{1}{2}} \right] \theta_n + |r|BN_{j-d}}{2 \left[\frac{1}{2}V\sigma^T + |\mu|A_{i+c} + \frac{1}{2}\Delta A(\beta - \mu) \right] N_{j-d}} \quad (3.39b)$$

and

$$1-c = \frac{SV\theta_s + \left[|\mu|A_{i-c}N_{i-c} + |r|BN_{j-d} \right] \theta_n + \frac{1}{2}\Delta A(\beta + \mu)N_{m-\frac{1}{2}}}{2 \left[\frac{1}{2}V\sigma^T + |\mu|A_{i+c} + |r|B \right] N_{m-\frac{1}{2}}} \quad (3.39c)$$

where θ_n and θ_s are arbitrary parameters between 0 and 1. It can be shown that the linear model corresponds to:

$$\frac{1}{a} = \frac{1}{b} = \frac{1}{c} = 2 \quad (3.40)$$

while the step model corresponds to:

$$\frac{1}{a} = \frac{1}{b} = \frac{1}{c} = 1 \quad (3.41)$$

As a practical matter, $\frac{1}{a}$, $\frac{1}{b}$, and $\frac{1}{c}$ are artificially limited between 1 and 2. This is highly important. In particular, it means that *when negative flux generation is not a possibility, the weighted model is identical to the linear model*. Thus, arguments as to which is more accurate in the small are moot. In fact, they are identical. In the case of large intervals, the weighting provides a smooth transition between the linear model and values near zero.

When $\beta = -\mu$, the equation for N is simplified as usual. Using Eqs. 3.36a and b together with Eq. 3.17:

$$N = \frac{VS + X'N_{i-c} + Y'N_{j-d}}{V\sigma^T + X' + Y'} \quad (3.42)$$

The coefficients a and b are derived in the previous manner, giving:

$$1-a = \frac{SV\theta_s + |\tau|BN_{j-d}\theta_n + |\mu|A_{i-c}N_{i-c}}{2 \left[\frac{1}{2}V\sigma^T + |\tau|B - \frac{1}{2}\Delta A\mu \right] N_{i-c}} \quad (3.43)$$

and

$$1-b = \frac{SV\theta_s + |\mu|A_{i-c}N_{i-c}\theta_n + |\tau|BN_{j-d}}{2 \left[\frac{1}{2}V\sigma^T + |\mu|A_{i+c} - \frac{1}{2}\Delta A\mu \right] N_{i-d}} \quad (3.44)$$

The c coefficient is not defined in this case, and is not used.

The parameters θ_s and θ_n are arbitrary, although it is intended that they lie between 0 and 1. Various references indicate a varying treatment over the years. The formulation discussed in the 1968 version of TWOTRAN (X,Y)¹⁷ is equivalent to:

$$\theta_s = \theta_n = 0 \quad (3.45)$$

This method never generates negative results, but will produce inaccurate results for large intervals with a source or for long, narrow intervals.

In 1969, a Lathrop paper¹⁸ had:

$$\theta_s = \theta_n = 1 \quad (3.46)$$

but testing with DOT has shown this method to lead to many near-zero degenerate results, thwarting convergence. In the 1973 version of DOT III,¹¹ a compromise:

$$\theta_s = 1 ; \quad \theta_n = 0 \quad (3.47)$$

was used. This worked well in source-dominated regions, but gave inaccurate results for source-free intervals of long, narrow shape. The "ordinary weighted-difference" model available in DOT 4 also has $\theta_s = 1$ and $\theta_n = 0$, but terms in Eq. 3.39 involving $N_{m-1/2}$ are multiplied by θ_s , rather than θ_n . This gives slightly better performance than the DOT III method without known penalties.

The " θ -weighted" model sets:

$$\theta_s = \theta_n = \theta \quad (3.48)$$

where θ is empirically chosen such as to avoid degeneracy, and yet to revert to the accurate linear model when the source or flow from adjacent boundaries will result in a good value of N . Values of θ ranging from 0.5 to 0.9 have been tested with satisfactory results.

Apparently any value near 1, but not near enough to produce numerical degeneracy, is acceptable. At present, a value of 0.9 is used in DOT 4. This θ -weighted model appears to be best for general-purpose application. It is not always the optimum model, however. If the problem is sufficiently simple, the linear-zero model may converge rapidly also, and it is less expensive per iteration. In a few very difficult problems, weighting coefficients can oscillate just as the zero fixup oscillated, with similar results. This oscillation is more likely with the θ -weighted treatment than with the ordinary weighted treatment.

Many more advanced formulations have been proposed in References 19, 20, 21, and elsewhere. For reasons which include cost, reliability, and limitations as to geometry, these methods have not yet been tested in DOT 4.

3.9. The Source Term

The source term in 3.3 includes, in general, sources from an extraneous (fixed) source distributed throughout the mesh, fission, scattering from other energy groups, and scattering from other directions within the same energy group. In principle, this can be expressed as:

$$S_{i,j,m,g} = Q_{i,j,m,g} + \chi_{i,j,g} \sum_{g'} \nu_{i,j,g} \sigma_{i,j,g'}^F \sum_{m'} W_{m'} N_{i,j,m',g'} + \sum_{g'} \sum_{m'} W_{m'} d_{i,j,m,g,m',g'} N_{i,j,m',g'} \quad (3.49)$$

The first and least controversial simplification is to group space cells having like material compositions into "material zones," and to require that all cells within such zones have the same cross sections. In principle, the cross sections might vary in a pointwise manner due to local temperature or density effects or to local differences in the spectrum used in obtaining group cross sections. In practice, such detail is rarely required, the detailed information basis is not available, and the mass of data would be too difficult to generate and store. In practice, density variation is handled by a separate density array, d_{ij} , which is understood to multiply all cross sections wherever they appear. Local temperature and weighting effects are represented by defining enough zones so that only cells of acceptably similar properties are grouped together. In what follows, the subscript z will be used to denote material zone, with the understanding that the zone number is a function of i, j . The implied factor $d_{i,j}$ will not be shown unless necessary to a particular argument.

A less satisfactory approximation is made in the fission term. In order to facilitate source iterations to resolve implicitness represented by the fission source, χ is assumed to depend only upon g . As a matter of convenience, ν and σ^F are written and treated as a single unit, since they depend upon the same variables. Thus, the fission term can be expressed as: $\chi_g D_{i,j}^F$, where:

$$D_{i,j} = \sum_{g'} \nu \sigma_{i,j,g'}^F \bar{N}_{i,j,g'} \quad (3.50)$$

$$\bar{N}_{i,j,g'} = \sum_{m'} W_{m'} N_{i,j,m',g'} \quad (3.51)$$

$D_{i,j}$ is termed "fission density," while $\bar{N}$ is "scalar flux." If several fissile species are present in the system, the use of a single χ_g for all space cells is not rigorously correct. Its use is justified largely by tradition and by experience, which has shown that most problems are not adversely affected by this treatment.

In principle, Eq. 3.49 can be used directly to calculate scattering. If, as discussed, zone cross sections are used:

$$\sum_{g'} \sum_{m'} W_{m'} \sigma_{z,m,g' n',g'} N_{i,j,m',g'} \quad (3.52)$$

and, in fact, certain transport studies have been performed in this way. In DOT, as in most contemporary discrete-ordinates codes, a moment expansion discussed by Mynatt³ is used. The directional dependence of σ^f in Eq. 3.1 is assumed to be represented satisfactorily by a Legendre polynomial in the single variable μ_0 , where:

$$\mu_0 = \underline{\Omega} \cdot \underline{\Omega'} \quad (3.53)$$

so that:

$$\sigma^f(\underline{r}, E, E', \underline{\Omega}, \underline{\Omega'}) \approx \frac{1}{4\pi} \sum_{l=0}^{l_{\max}} P_l(\mu_0) \sigma_l^f(\underline{r}, E, E') \quad (3.54)$$

and in Eq. 3.3, the corresponding representation is:

$$\sigma_{z,m,g' n',g'}^f \approx \frac{1}{4\pi} \sum_{l=0}^{l_{\max}} P_l(\mu_0) \sigma_{l,z,g,g'}^f \quad (3.55)$$

where μ_0 is now the cosine of the angle between directions m and m' . In a difficult problem, $l_{\max}$ may be 3, while m and m' may run to 200 each. Thus, 40,000 values are approximated by 3. Also, the σ_l^f are independent of the direction set. A single set of cross section data can be stored and used with many different direction sets. It should be noted that cross sections used by LANL codes traditionally have a factor of $2l + 1$ included in Eq. 3.53, which results in values σ_l^f which are a factor of $2l + 1$ lower. ORNL codes do not assume this factor. Thus, LANL-type cross sections must be multiplied by $2l + 1$ before use in DOT 4.

At this point, it is fruitful to apply a corresponding expansion to ψ . Since ψ is inherently a function of the full direction space, rather than of one angle, a type of spherical harmonic expansion is used.

We now define:

$$C_l^k(\mu, \phi) = \left[\frac{2l+1}{4\pi} \frac{(2-\delta_{k0})(l-K)!}{(l+K)!} \right]^{1/2} P_l^K(\mu) \cos K \phi \quad (3.56)$$

where δ_{K0} is the "Kronecker delta," 1 when $K = 0$ and 0 otherwise, and the P_l^K are the associated Legendre polynomial basis functions as shown on the previous page. In cylindrical geometry, if $\underline{i}$, $\underline{j}$, $\underline{k}$ are unit vectors along the R, Z, and θ coordinate axes, ϕ is the angle which the projection of the direction of travel in the plane formed by $\underline{j}$ and $\underline{k}$ makes with $\underline{j}$. Its analogous definition in other geometries is illustrated in Reference 2. With these, ψ is approximated:

$$\psi(\underline{r}, E, \Omega) \approx \sum_{l=0}^{l_{\max}} \sum_{k=0}^l C_l^k(\mu, \phi) \psi_l^k(\underline{r}, E) \quad (3.57)$$

In the finite difference form, the function C_l^k is replaced by an array of discrete values:

$$C_{l,m}^K \equiv C_l^K(\mu_m, \phi_m) \quad (3.58)$$

and the corresponding flux approximation is:

$$N_{i,j,m,g'} \approx \sum_{l=0}^{l_{\max}} \sum_{k=0}^l C_{l,m}^K N_{l,j,g}^* \quad (3.59)$$

The advantage of this is twofold. First, as many as 200 flux values in the case cited can be approximated by 10 stored moments. Also, it can be shown that the scattering portion of the source term can be obtained directly from the moments:

$$\sum_{g'} \sum_{m'} W_{m'} \sigma_{i,j,m,g,m',g'}^* N_{i,j,m',g'} \approx \sum_{g'} \sum_l \sum_k \sigma_{l,z,g,g'}^* N_{l,j,g}^* \quad (3.60)$$

As a matter of convenience, the extraneous source is similarly expressed:

$$Q_{i,j,m,g} \approx \sum_{l=0}^{l_{\max}} \sum_{k=0}^l C_{l,m}^K Q_{l,j,g}^* \quad (3.61)$$

and is called a "distributed source" in this form. In actual calculations, both Q and N are supplied in moment form as input. The source in a given direction is easily found using:

$$S_{i,j,m,g} \approx \chi_g D_{i,j} + \sum_{l=0}^{l_{\max}} \sum_{k=0}^l C_{l,m}^K \left[Q_{l,j,g}^* + \sum_{g'} \sigma_{l,z,g,g'}^* N_{l,j,g}^* \right] \quad (3.62)$$

As new N_m 's are calculated, the moments are calculated by:

$$N_{l,j,g}^* = \sum_m W_m C_{l,m}^K N_{i,j,m,g} \quad (3.63)$$

and stored. This allows N_m 's to be discarded at once, freeing enormous amounts of data storage in a large problem. It may also be noted that group g' need not have the same direction set as group g when the moment representation is used.

The expansion represented by Eq. 3.59 is not without its problems. If $l^{\max}$ is too small, details of the flux shape are lost. If scattering is highly anisotropic, then, the scattering term in Eq. 3.62 will not be correct. On the other hand, the theory which allows evaluation of the N_l^K by Eq. 3.63 assumes that integration over all direction space will be performed, in which case the $C_l^K(\mu, \phi)$ are orthogonal. When the discrete quadrature of Eq. 3.63 is used, the corresponding orthogonality condition is satisfied only when $l^{\max}$ is sufficiently small.

In addition, the cross section expansion of Eq. 3.54 can badly misrepresent certain cross sections, resulting in meaningless oscillation between positive and negative scatters. Cross sections representing high-energy Compton scattering are an example of this. The DOT 4 code has a special feature which partially mitigates the effects of this difficulty.

In spite of these difficulties, the " $P_L S_N$ " method described here has been proven in much experience and in many comparisons with experiment. It is nearly universally accepted.

3.10. Directional Quadrature Sets

In the method of discrete ordinates represented herein, integrals over direction space are replaced with weighted sums over sets of discrete directions. Although the directional weights may be thought of as representing areas on a unit sphere adjacent to the discrete directions, no attempt is made to define those areas. Instead, the directions and weights are chosen in such a way as to provide as much accuracy as possible in the quadrature.

The directions are represented by their direction cosines (μ, η, ξ) with respect to a set of orthogonal coordinate axes. With $\underline{i}$, $\underline{j}$, $\underline{k}$ as unit vectors along the axes, the discrete direction set is:

$$\mu_m \underline{i} + \eta_m \underline{j} + \xi_m \underline{k} \quad (3.64)$$

We refer to the sets of cosines, together with the corresponding weights, W_m , as "discrete-ordinates directional quadrature sets." Integrals are then represented as, for example:

$$\int f(\underline{\Omega}) d\underline{\Omega} \cong \sum_m W_m f(\mu_m, \eta_m, \xi_m) \quad (3.65)$$

Two constraints on the sets are immediately apparent:

$$\sum_m W_m = 1 \quad (3.66)$$

$$\mu_m^2 + \eta_m^2 + \xi_m^2 = 1 \quad (3.67)$$

It is also apparent that one wishes the first moments of a symmetrical function to be 0, so that isotropic flux will have no current:

$$\sum W_m \mu_m = 0 \quad (3.68a)$$

$$\sum W_m \eta_m = 0 \quad (3.68b)$$

$$\sum W_m \xi_m = 0 \quad (3.68c)$$

Additional constraints are added as the number of degrees of freedom increases, in order to provide maximum accuracy. Jenal et al.,²² for example, discuss this matter in more detail.

Proper ordering of the directions is essential. For the DOT code and certain other codes, all directions with common η must be grouped together in order of increasing μ . Each such group (called an η -level) must have, as its first member, an arbitrary boundary direction for which $\xi = 0$. Thus, for a given η , the boundary direction has:

$$\mu = -\sqrt{1 - \eta^2} \quad (3.69)$$

These directions are assigned weights of 0. All directions having $\eta < 0$ must precede directions with $\eta > 0$.

In a two-dimensional code, only 4 octants of a unit sphere are considered. The other four are redundant, and are included implicitly in the weight normalization. Some DOT 4 options also bypass the calculation of results in the 0-weight directions. Even so, the directions must remain in the direction sets for proper functioning of the code.

With XZ and RZ symmetry, only μ and η are supplied to the code. With R θ symmetry, μ and ξ are supplied. Even so, the ordering of the direction sets remains unchanged. Zero values of ξ must be replaced by small non-zero values for proper functioning of the coding. Also, ξ must have the same sign as the corresponding η .

Sets can be "unbiased" or "asymmetric," i.e., in an RZ calculation, if (μ, η) is in the set, then $(-\mu, \eta)$ and $(\mu, -\eta)$ are also in the set, or "biased," in which many additional directions provide fine detail of streaming features.

3.11. Internal Boundary Sources

For certain applications, sources may be required at cell boundaries. This obviously can not be represented by Eq. 3.62. If we represent such a source as, for example, a source at the $i + 1/2$ boundary as $S_{i+1/2,j,m,\xi}^B$, then the effect of the source is to make the flux at one side of the boundary larger than the flux at the other side by the amount S^B . That is, when i' is the successor of interval i "following the flow," i.e., $i' = i + 2c$, N_{i+c} the boundary flux leaving interval i , and $N_{i'-c}$ is the boundary flux used in calculating $N_{i'}$, then the effect of S^B is:

$$N_{i'-c} = N_{i+c} + S_{i+c}^B \quad (3.70)$$

3.12. External Boundary Conditions

In previous section, it was noted that the value of N must be artificially set at each bound. according to the physical conditions at that boundary. The most common condition is "void," or "vacuum." It is effected by setting all inward-directed fluxes to 0.

The other common condition is "reflected." This condition describes the condition at a plane of symmetry, such as that at the horizontal midplane of a homogeneous cylinder. The reflected condition is also frequently used at the center of cylinders and at the outer surface of cylinders which represent extended lattices, such as mockups of fuel rods in a regular array. It is effected by setting the inward flux equal to the outward flux in the "reflected direction." For example, at a horizontal top boundary, flux entering in direction $(-\mu, -\eta)$ is set equal to the flux emerging in direction $(-\mu, \eta)$ or, at least, to the best available estimate of it. Iteration strategy is to be discussed later. At a vertical right-hand boundary, flux entering in direction $(-\mu, -\eta)$, for example, is set to the flux emerging in direction $(\mu, -\eta)$.

The "albedo" or "grey" boundary condition adjusts incoming flux such it is isotropic, and such that the ratio of incoming current to outgoing current is a constant called the albedo, α . For example, at a top boundary, downward flux is set to:

$$N_m = \alpha \frac{\sum_{m'} W_{m'} N_{m'} ; \eta_{m'} > 0}{\sum_{m'} W_{m'} ; \eta_{m'} < 0} \quad (3.71)$$

This condition is often used in shielding work and air transport problems to give an approximate representation of material for which a detailed calculation is not required. It is also used to allow representation of control rod effects, the details of which have been calculated by another process. A "white" boundary condition is simply the special case of $\alpha = 1$. It is sometimes used at the outer boundary of cells, but with poor results.

The cylindrical boundary condition, when applied to a vertical boundary, has the effect of a white boundary condition applied by η -level, i.e.,

$$N_{\eta} = \frac{\sum_{m'} W_{m'} N_{m'} ; \mu_{m'} \mu_m < 0 ; \eta_{m'} = \eta_m}{\sum_{m'} W_{m'} ; \mu_{m'} \mu_m > 0 ; \eta_{m'} = \eta_m} \quad (3.72)$$

This has been shown to be much better than any of the preceding conditions in representing a regular array of cells when used at the outer boundary of a cylinder.²³ It is also theoretically more appropriate along the centerline of a cylinder, since it assures that outward flux will be truly independent of μ . Its results are typically equivalent to those of a reflected condition at the centerline of practical problems, however.

When applied at a horizontal boundary, the cylindrical condition allows a degree of asymmetry:

$$N_m = \frac{\sum_{m'} W_{m'} N_{m'} ; \eta_{m'} = -\eta_m}{\sum_{m'} W_{m'} ; \eta_{m'} = \eta_m} \quad (3.73)$$

In theory, this allows reflection at a vertical boundary at which the μ 's for given η pairs do not match. It has not received much practical testing, however.

The "periodic" boundary condition represents a continuing array of asymmetric objects, e.g. an ...ABCCABCC... array. It is applied by setting the ingoing flux at a boundary to the flux in that same direction at the opposite boundary. As an example, recalling that $i = 1 + 1/2$ is a right boundary opposite the left boundary $i = 1/2$:

$$N_{1+1/2,j,m,g} = N_{1/2,j,m,g} ; \mu_m < 0 \quad (3.74)$$

The periodic boundary condition is the correct condition for the top and bottom of a 360° circular mesh, described later.

The "fixed source" or "flux" boundary condition simply sets the flux to a value specified as input. It is most often used in problems which are a spatial continuation of previous problems. An option allows flux at an interval boundary to be received as output from one problem and then to be used as an "external boundary source" at a fixed-source boundary.

3.13. Iteration Strategy

Two of the chief advantages of discrete-ordinates theory over diffusion theory and similar methods are that the calculation of flux is fully explicit for a given source, thus not requiring spatial iterations, and that the details of directional flux distribution are treated. Unfortunately, one takes away what the other gives. Since the source involves an integral over flux moving in other directions within the same energy group, iterations over source must be used. The scattering integral is evaluated based on prior information or guesses as available, all of the fluxes for an energy group are evaluated, a new scattering integral is evaluated, and so on. From 4 to 25 of these "inner" or "flux" iterations may be required for typical problems. The iterations are deemed to have "converged" when the fluxes from two or more successive iterations are sufficiently close to each other. As discussed later, powerful means of accelerating these iterations are required with difficult problems.

It may be noted that reflected, albedo, or cylindrical boundary conditions at the top or right-hand boundary of a mesh also require iterations. Periodic boundary conditions at any boundary similarly require iterations.

Problems which have no fission and no "upscatter," i.e., no scattering from g' to g where $g' > g$, can be solved with one pass through the energy groups, and it is inefficient to do more. If either fission or upscatter are present, however, another level of implicitness in the source term is created. The source for such a problem depends upon fluxes which have not yet been calculated in the sweep of energy groups. This is solved by estimating the fission and upscatter contribution from previous information or from the input flux guess, and then evaluating flux, a new source, a new flux, etc. These "source" or "outer" iterations must also be accelerated in order to solve difficult problems.

It is possible, and sometimes economical, to perform only 1 flux evaluation at each pass through the groups if several source iterations are to be performed, and satisfactory computer codes have been constructed to operate in this way. Studies with DOT show that 2 to 5 flux iterations per source iteration give optimum results for many problems. The

number of source iterations required may vary from as few as 4 to 100 or more, depending on the problem, the success of the acceleration procedures, and the degree of convergence required.

3.14. Fission Problems, K-Calculations, and Searches

If a fixed source (distributed, internal boundary, or external boundary) is present, Eqs. 3.3 and 3.62 can be iterated to a stable solution, providing $\nu\sigma^F$ is not so large as to make the system critical. As the system nears critical, the number of progeny produced by a single source particle (called "multiplication," M) becomes very large. Convergence becomes difficult and dependent upon computer precision. Convergence for $M > 100$ is difficult, for example, on an IBM 370 having 10^{-6} word precision. As $\nu\sigma^F$ increases, M approaches ∞ , and no stable solution can be reached.

Without a fixed source, however, meaningful solutions are obtained by searching for the single parameter (the "multiplication factor," K_{eff}) by which χ must be divided in order to bring Eqs. 3.3 and 3.62 to a stable solution. DOT 4 uses a very simple procedure for the basic evaluation of K_{eff} . The initial source is calculated based upon the flux guess. The fluxes are normalized such that the fission component of the source is 1. New fluxes and a new fission source are calculated. The ratio of new fission source to old is taken as the estimate of K_{eff} . Sources and fluxes are renormalized as before, and the process repeats until both the fission source and K_{eff} estimate converge. Thus, a new estimate of K_{eff} is obtained with each source iteration. Techniques are available to accelerate the convergence by replacing source and flux from one iteration with extrapolated results which are nearer the ultimate solution.

Three types of search options are available. The buckling search ("B") multiplies an input value of " DB^2 " by an adjustable parameter (the "eigenvalue," λ) until the resulting value of K_{eff} , evaluated as described earlier, is a specified value. The effect of the buckling is simply to add the amount $\lambda(DB^2)$ to σ^T everywhere. The concentration search ("C") adjusts λ such as to modify the cross sections for certain specified constituents in the proper amount to achieve the specified value of K_{eff} . The dimension search ("D") modifies λ such as to find new widths for specified mesh intervals which achieve the specified value of K_{eff} .

The search for λ is conducted in the same manner in each case. The value of λ is set to an initial value, λ_1 , and K_1 , a partially-converged value of K_{eff} is obtained. A new value, λ_2 , is selected based on both input data and K_1 , and K_2 is obtained. At this point, a nonlinear procedure which tends to prevent overextrapolation is used to estimate λ_3 , from which K_3 is evaluated. From this point onward, the newest three values of λ and estimates of K are used to extrapolate until K is sufficiently near the specified value.

The three-point extrapolation uses a fitting function suggested by D. R. Vondy,²⁴ where A , B , and C are arbitrary fitting parameters:

$$\lambda = C + AK/(B-K) \quad (3.75)$$

If only 2 points are available for the fit, C is arbitrarily assumed to be 0. If a negative λ should result, or if B should be larger than 10^4 , the nonlinear fit is considered invalid.

and a linear fit is used. With either linear or nonlinear extrapolation, an oscillation test is applied. If the trend of λ has reversed signs, i.e., if:

$$(\lambda_N - \lambda_{N-1}) (\lambda_{N-1} - \lambda_{N-2}) < 0 \quad (3.76)$$

then oscillation damping is applied. Where $\tilde{\lambda}_{N+1}$ is the new estimate of λ calculated from Eq. 3.75, a modified λ_N is tried:

$$\lambda_{N+1} = \lambda_N + \gamma (\tilde{\lambda}_{N+1} - \lambda_N) \quad (3.77)$$

The value of γ is presently set at 0.8.

As an additional protection against meaningless extrapolation, λ is allowed to vary only within an input factor of its original value. If K departs from the objective value by more than a specified value, or if λ is "stuck" on the same value for two successive iterations, the search is halted. The relative change in λ per iteration is required to be no more than a specified value.

This is certainly not a highly sophisticated scheme. It will obviously be defeated if the variation allowed in λ is not sufficient to adjust K to the desired value. If an input error causes λ to have no effect at all, a perplexing, aimless drift will be observed. The input includes a "convergence ratio" which allows λ to change when K is only partially converged. An unduly low value of the ratio can cause premature extrapolation of λ and subsequent poor results. In general, however, the scheme has worked well.

3.15. Rebalance and Acceleration

3.15.1. Groupwise Rebalance

The flux iterations in a typical problem must be accelerated in some way in order to bring the cost of difficult problems within acceptable limits. The oldest and simplest method is "group rebalance." First, let us split the source term of Eq. 3.49 into two parts, separating the scattering portion which originates within the energy group, itself, the "self scatter" (H_g) from the remainder of the source (G_g):

$$G_g = \sum_i \sum_j V_{ij} x_g D_{i,j} + \sum_m W_m \left[Q_{m,g} + \sum_{g' \neq g} \sum_{m'} W_{m'} \sigma_{m,g,m',g'} N_{m',g'} \right] \quad (3.78)$$

$$H_g = \sum_i \sum_j V_{ij} \sum_m W_m \sum_{m'} W_{m'} \sigma_{m,g,m',g} N_{m',g} \quad (3.79)$$

Now, let us integrate Eq. 3.3, as it applies to an individual energy group, over direction and space. The result can be expressed as:

$$\bar{L}_g + \bar{T}_g = \bar{Q}_g + \bar{H}_g \quad (3.80)$$

where we use the double bar to represent space-and-direction integrals. $\bar{\bar{L}}_g$ represents total boundary leakage, and $\bar{\bar{T}}_g$ total removal due to the σ^T term. Now, let us consider the actual iteration cycle. For the p^{th} flux iteration, $\bar{H}_g$ must be based upon out-of-date information from iteration $p-1$. Then, $\bar{\bar{L}}_g$ and $\bar{\bar{T}}_g$ obtained by applying Eq. 3.3 to $\bar{H}^{p-1}$ will not truly satisfy the intended equation. Let us designate the result of such application with a tilde. Then:

$$\tilde{\bar{\bar{L}}}_g + \tilde{\bar{\bar{T}}}_g = \bar{G}_g + \bar{H}_g^{p-1} \quad (3.81)$$

when $\bar{H}_g^p$ is calculated from the p^{th} iteration fluxes, it will be different, and will give rise to still different fluxes in the $(p+1)^{th}$ iteration. If $\bar{H}_g$ is, say, 99% of $\bar{\bar{L}}_g + \bar{\bar{T}}_g$, then the cycle converges very slowly. The intent of rebalance is to use an inexpensive process to estimate the end of the iteration cycle, and then to use that estimated result in the next iteration. Assume an arbitrary factor, f , by which all fluxes, $\bar{N}^p$ are to be multiplied in order to bring fluxes and sources into balance. Terms L , T , and H will be proportional to f , but G will not:

$$f_g \tilde{\bar{\bar{L}}}_g + f_g \tilde{\bar{\bar{T}}}_g = \bar{G}_g + f_g \bar{H}_g^p \quad (3.82)$$

Solving Eqs. 3.81 and 3.82 to eliminate L and T :

$$f_g = \frac{\bar{G}_g}{\bar{G}_g - (\bar{H}_g^p - \bar{H}_g^{p-1})} \quad (3.83)$$

Then, the $(p+1)^{th}$ iteration is started with:

$$N_g^p = f_g \bar{N}_g^p \quad (3.84)$$

This simple procedure is quite helpful for solving problems in which flux shape is dominated by leakage, such as small metal assemblies. It is not applicable to large problems in which changes in flux shape, rather than changes in overall magnitude, dominate convergence.

3.15.2. Upscatter Rebalance

A similar situation exists with respect to convergence of problems having upscatter. If we sum the terms of Eq. 3.82 over those groups into which upscatter is present, we may regard the sum as a "super group." Scattering between groups of the super group would be included within the self-scatter term, H , of such a super-group. Suppose all groups $g > g'$ have scattering from lower groups into them. Then, with s designating the super-group:

$$\begin{aligned} \bar{G}_s = & \sum_i \sum_j v_{ij} \sum_{g=g'+1}^{g''} \left\{ \chi_g D_{ij} + \sum_m Q_{m,g} \right. \\ & \left. + \sum_{g'=1}^{g'} \sum_{m'} \sigma_{s,m,g,m',g'} N_{j,m',g'} W_{m'} \right\} W_m \end{aligned} \quad (3.85)$$

$$\bar{H}_s = \sum_i \sum_j \nu_{ij} \sum_{g=g'+1}^{\infty} \sum_m \sum_{m'} \sigma_{i,j,m,g,m',g} N_{i,j,m,g} W_m W_{m'} \quad (3.86)$$

The relationship of flux to source for this super-group is entirely analogous to the groupwise scaling situation. A similar derivation leads to a scaling factor for the upscatter super-group which is to be applied between source iterations (designated by q here):

$$f_s = \frac{\bar{G}_s}{\bar{G}_s - (\bar{H}_s^q - \bar{H}_s^{q-1})} \quad (3.87)$$

The combination of groupwise and upscatter rebalance insure that the results of a source iteration are always in full balance, at least in a global sense. Total particle conservation is thus insured to full computer accuracy, and many problems can be solved satisfactorily with only these accelerations.

3.15.3. Space Rebalance

By 1968, early versions of DOT were being applied to deep-penetration transport problems, and groupwise rebalance was no longer adequate. From similar considerations, applied on a mesh-cell basis, "space rebalance" was developed.²⁵ Integrating Eq. 3.3 over direction, but not over space, and defining "partial currents":

$$F_{i+1/2} = A_{i+1/2} \sum_m N_{m,i+1/2} \mu_m W_m ; \quad \mu_m > 0 \quad (3.88a)$$

$$F_{i,j+1} = A_{i+1/2} \sum_m N_{m,i+1/2} \mu_m W_m ; \quad \mu_m < 0 \quad (3.88b)$$

$$F_{j+1/2} = B_{j+1/2} \sum_m N_{m,j+1/2} \eta_m W_m ; \quad \eta_m > 0 \quad (3.88c)$$

$$F_{j,j+1} = B_{j+1/2} \sum_m N_{m,j+1/2} \eta_m W_m ; \quad \eta_m < 0 \quad (3.88d)$$

"inscatter source":

$$\bar{G}_{i,j,g} = \chi_g D_{i,j} + \sum_m \left[Q_{i,j,m,g} + \sum_{g' \neq g} \sum_{m'} N_{i,j,m',g'} \sigma_{i,j,m,g,m',g} W_{m'} \right] W_m \quad (3.89)$$

"scalar self scatter":

$$\bar{\sigma} = \sum_m \sum_{m'} \sigma_{m,g,m',g} W_m W_{m'} \quad (3.90)$$

and "removal":

$$\sigma^R = \sigma^T - \sigma^s \quad (3.91)$$

yields, where we use tilde to denote an unbalanced result:

$$\begin{aligned} \tilde{F}_{i+1,i} - \tilde{F}_{i,i+1} + \tilde{F}_{i-1,i} - \tilde{F}_{i,i-1} + \tilde{F}_{j+1,j} - \tilde{F}_{j,j+1} + \tilde{F}_{j-1,j} - \tilde{F}_{j,j-1} \\ + v(\sigma^R + \sigma^s)\tilde{N} = v\bar{G} + v\sigma^s\tilde{N} \end{aligned} \quad (3.92)$$

In the discussion which follows, we drop the j -flows, giving a 1-D result, for the sake of simplification. The missing terms can easily be added by inspection. As before, the flux, N^p , from the p^{th} iteration does not truly satisfy Eq. 3.92, since $\bar{N}^{p-1}$ must be used in calculating the source. In actual fact:

$$- (\tilde{F}_{i,i+1} + \tilde{F}_{i,i-1}) + (\tilde{F}_{i+1,i} + \tilde{F}_{i-1,i}) + v(\sigma^R + \sigma^s)\tilde{N}^p = v\bar{G} + v\sigma^s\bar{N}^{p-1} \quad (3.93)$$

As before, let us define rebalance factors which will bring about cellwise balance with the p^{th} source. Let us assume that partial flows are proportional to the factor for the cell from which they emerge. With these factors in hand, the rebalanced fluxes will be:

$$\tilde{N}_{i,j,s}^p = \tilde{N}_{i,j,s}^p f_{i,j,s} \quad (3.94)$$

In these terms, the balanced flux satisfies:

$$\begin{aligned} - (\tilde{F}_{i,i+1}\bar{N}_{i+1}^p/\tilde{N}_{i+1}^p + \tilde{F}_{i,i-1}\bar{N}_{i-1}^p/\tilde{N}_{i-1}^p) \\ + \left[(\tilde{F}_{i+1,i} + \tilde{F}_{i-1,i})/\tilde{N}_i^p + v(\sigma^R + \sigma^s) \right] \bar{N}_i^p \\ = v\bar{G} + v\sigma^s \bar{N}_i^p \end{aligned} \quad (3.95)$$

It is instructive to eliminate terms in $\bar{G}$, using Eqs. 3.93 and 3.95. The result can be expressed:²⁶

$$\sum_i Z_{i,j} \bar{N}_i^p = \sum_i Z_{i,j} \bar{N}_i^{p-1} + v\sigma^s(\bar{N}_i^p - \bar{N}_i^{p-1}) \quad (3.96)$$

where

$$Z_{i,j'} = - \tilde{F}_{i,j'}^p / \tilde{N}_i^p ; \quad i' = i \pm 1 \quad (3.97a)$$

$$Z_{i,j'} = v\sigma^R + \left(\tilde{F}_{i+1,j}^p + \tilde{F}_{i-1,j}^p \right) / \tilde{N}_i^p ; \quad i' = i \quad (3.97b)$$

$$Z_{i,j'} = 0 ; \quad i' \neq i \pm 1 \quad \text{and} \quad i' \neq i \quad (3.97c)$$

This can be solved by standard matrix-solution routines, producing a new set of fluxes, $\tilde{N}_i^p$, which are in balance with the p^{th} source.

Expressed in this form, the method can be seen to be an application of the method of synthesis propounded by Kopp²⁷ and explored by Gelbard and Hageman.²⁸ The method calculates, in effect, the result which would be obtained if no directional redistribution of flux were required in subsequent transport iterations. The directional flux can be "fixed up" such as to correspond to $\tilde{N}_i^p$ either by multiplying all directional fluxes at position i by a constant or by adding a constant to the fluxes. DOT uses the multiplicative method, which preserves the original directional distribution. The additive method can produce negative fluxes where none existed before, and it is not evident how this difficulty could be overcome.

Proper boundary conditions for $\tilde{N}^p$ must be set. Flow from a fixed external boundary source belongs with the G term, and does not appear in Eq. 3.97. Thus, F at such a boundary is set to 0. For void, reflected, albedo, and cylindrical boundaries, it is recognized that the inward flow at the left boundary is α times the outward flow, where $\alpha = 0$ for a void, 1 for reflected or cylindrical conditions, and $\alpha = \text{albedo}$ for an albedo boundary. This is implemented by setting $Z_{1,0}$ to 0 and adding $-\alpha \tilde{F}_{0,1}^p / \tilde{N}_1^p$ to $Z_{1,1}$. A similar matrix adjustment describes the periodic condition. An analogous treatment has often been used at the right boundary, also, but it is not rigorously correct.²⁹

In the case of a right boundary, inward flow, represented by $\tilde{F}_{I,J+1}^p$, is not proportional to $\tilde{N}_I^p$ at all. In fact, it is $\alpha \tilde{F}_{I+1,J}^p$.

A treatment exactly analogous to the left-boundary treatment would set $Z_{I,J+1}$ to 0 and would replace $\tilde{F}_{I+1,J}^p$ in Eq. 3.97b with $(1-\alpha)\tilde{F}_{I+1,J}^p$. To be entirely consistent, however, a new term must be added to the righthand side of Eq. 3.96:

$$\begin{aligned} \sum_{i'} Z_{i,j'} \tilde{N}_i^p &= \sum_{i'} Z_{i,j'} \tilde{N}_i^{p-1} + v\sigma^R \left(\tilde{N}_i^p - \tilde{N}_i^{p-1} \right) \\ &+ \left[\alpha \tilde{F}_{I+1,J}^p - \tilde{F}_{I,J+1}^p \right] ; \quad i=I \end{aligned} \quad (3.98)$$

Thus, residual boundary inflow has an effect entirely analogous to residual scattering. The boundary residual term has a beneficial effect which ranges from small to very large, according to the importance of the boundary and the speed of its convergence.

An alternate form of Eq. 3.96 is also available. For this, we eliminate terms in σ^R from Eqs. 3.93 and 3.95, rather than terms in $\bar{G}$:

$$\begin{aligned}
 & - \left(\tilde{F}_{i,j+1}^{p} f_{i+1} + \tilde{F}_{i,j-1}^{p} f_{i-1} \right) \\
 & + \left[\tilde{F}_{i+1,j}^{p} + \tilde{F}_{i-1,j}^{p} + v\bar{G} - \sigma^p \left(N_i^p - N_i^{p-1} \right) \right] f_i = v\bar{G} \quad (3.99)
 \end{aligned}$$

In principle, the results of Eq. 3.99 must be identical with those of Eqs. 3.94 and 3.96. In practice, the matrix of Eq. 3.99 is easier to solve.

3.15.4. Damped Partial Current Rebalance

It is evident from Eq. 3.99 that negative or divergent f_i 's may result if σ^p is too large. In fact, the application of that method was so troubled with instability that it was customary to perform one or two unbalanced iterations after each rebalanced iteration.

Reed showed that the stability problem could be cured by use of fictitious boundary currents which tend to "damp" the rebalance, i.e., to drive the factors toward unity in areas where they are likely to be erratic.³⁰ In Reed's method, the boundary flows are calculated as usual. Then, each is augmented with a fictitious flow as follows, where the prime denotes the stabilized flow:

$$\tilde{F}_{i,j+1}' = \tilde{F}_{i,j+1} + F_{i+\frac{1}{2}}^B \quad (3.100a)$$

$$\tilde{F}_{i+1,j}' = \tilde{F}_{i+1,j} + F_{i+\frac{1}{2}}^B \quad (3.100b)$$

Some ambiguity is present in the general definition of F^B , since Reed's work was for slab geometry with uniform mesh spacing. DOT 4 allows two definitions,

" Φ Method":

$$F_{i+\frac{1}{2}}^B = \frac{z'}{4} \max \left\{ v_i \sigma_i^p N_i, v_{i+1} \sigma_{i+1}^p N_{i+1} \right\} \quad (3.101a)$$

"J Method":

$$F_{i+\frac{1}{2}}^B = z' \max \left\{ \tilde{F}_{i+1,j}', \tilde{F}_{i,j+1}' \right\} \quad (3.101b)$$

with either of these definitions, Eq. 3.99 becomes:

$$\begin{aligned}
 & - \left(\tilde{F}_{i,j+1}' f_{i+1} + \tilde{F}_{i,j-1}' f_{i-1} \right) \\
 & + \left[\tilde{F}_{i+1,j}' + \tilde{F}_{i-1,j}' + v\bar{G} - \sigma^p \left(N_i^p - N_i^{p-1} \right) \right] f_i = v\bar{G} \quad (3.102)
 \end{aligned}$$

and this is the "damped partial-current rebalance" (PCR) used in DOT 4.

In 1-D geometry, Reed's stability criterion is equivalent to $z' = 1$. A later paper showed that non-negativity could be assured by $z' = 2$.³¹ In 2-D geometry, $z' = 1$ assures non-negativity. It has been found that allowing z' to increase as the iterations progress can extend the usefulness of the method to much more difficult problems. The current version of DOT 4 allows the user to input an initial value of z' , and an increment, $\Delta z'$, which is to be applied after rebalance.

The choice of $\Delta z'$ is highly influential in determining the solution time of a problem. A non-optimum choice may double the time, or even cause failure. The optimum value is, unfortunately, problem dependent. Partial relief to this dilemma has been added in the form of self-adjusting damping.²⁸

Defining, on a pointwise basis:

$$e = (\tilde{N}^p - N^{p-1})/\tilde{N}^p \quad (3.103)$$

$$f = (\bar{N}^p - \tilde{N}^p)/\tilde{N}^p \quad (3.104)$$

$$\hat{e} = \text{value of } e \text{ where } |e| \text{ is maximum} \quad (3.105)$$

$$\hat{f} = \text{value of } f \text{ where } |f| \text{ is maximum} \quad (3.106)$$

Then the increment $\Delta z'$ is applied if and only if:

$$\hat{f} \hat{e} < 0 \quad (\text{rebalance is opposing iteration}) \quad (3.107)$$

$$\text{or } \hat{f}^p \hat{f}^{p-1} < 0 \quad (\text{rebalance is oscillating}) \quad (3.108)$$

$$\text{or } |\hat{f}^p| > |\hat{f}^{p-1}| \quad (\text{rebalance is diverging}) \quad (3.109)$$

This procedure allows a value of $\Delta z'$ which is large enough for pathological problems to be built into the code without harming the execution of normal problems greatly.

In spite of the awkwardness of the damping selection, PCR remains the workhorse of deep penetration problems, especially at high energy. It is sometimes ineffective at highly-scattering epithermal and thermal energies, and may fail entirely in such cases.

3.15.5. Diffusion Acceleration

Alcouffe showed that the difficulties of PCR could be avoided by use of an entirely different form of synthesis, which he called "diffusion synthetic acceleration" (DSA).³² He showed three different means of using a modified diffusion equation as the "less expensive" operator in a synthesis process. His forms were not directly useful to DOT, but they soon led to "consistent diffusion acceleration" (CDA) which was useful.³³ In essence, CDA

simply involves replacing the flow terms of Eq. 3.88 with terms calculated from diffusion theory. In slab geometry:²⁶

$$C_{i,j'} = -D^T A / \Delta x \quad ; \quad i' = i \pm 1 \quad (3.110a)$$

$$C_{i,j'} = v \sigma^R - \left(C_{i+1,j} + C_{i-1,j} \right) \quad ; \quad i' = i \quad (3.110b)$$

$$C_{i,j'} = 0 \quad ; \quad i' \neq i \pm 1 \quad \text{and} \quad i' \neq i \quad (3.110c)$$

where D^T is the diffusion coefficient, and A is the area of the face between cells i and j . Unfortunately, this formulation is unstable without additional help, just as space rebalance was. Reed's method has no application here. Two procedures have been successful in suppressing the instability, and neither introduce a new difficulty such as the introduction of the z^* parameter.

First, the rebalance flux and rebalance factors can be calculated on the corners of the mesh cells (CNRDA), and then corner values can be averaged for application to the mesh cells as before. Alcouffe used an approach similar to this, and it is entirely successful. The use of mesh corners, however, introduces fairly complex coding. Second, the rebalance factors can be calculated at the mesh centers as indicated by Eq. 3.110, and then the mesh-corner values can be deduced by averaging adjacent mesh centers (CTRDA). The resulting center-corner-center transformation amounts to a 5-point smoothing process which also brings stability to the CDA method. In many tests, CTRDA has performed well, but never as well as CNRDA.

The present CNRDA routines bring quick convergence to just those highly-scattering low-energy groups which PCR cannot solve. It is sometimes slightly less efficient at high energies, where direct propagation of the source, rather than scattering, is the predominant action and convergence may occasionally not be obtained on difficult problems having highly anisotropic effects. Even so, when all energy groups are considered, CNRDA is usually much faster than PCR. The details of both CNRDA and CTRDA are too extensive for presentation here.

3.15.6. Source Iteration Acceleration

In principle, the problem of accelerating source iterations is comparable to the flux-iteration problem. The calculation must start with a source which results in new fluxes which result in a somewhat different source. The concept of synthesis appears equally applicable here, but the coding required to achieve it is more complex. The DOT III W code reported a synthesis-style rebalance in the energy variable,³⁴ but did not revise the spatial shape. The TWOTRAN-II code solves the alternate problem - an energy-independent space rebalance.² We do not know of applications of a full space-energy synthesis to the outer iteration problem, although it seems to be an attractive option.

DOT 4 uses three types of source-iteration acceleration. The first, "fission rescaling," is used when a fixed-source problem also has a fission component. It is entirely analogous to

groupwise and upscatter scaling. Where $\bar{Q}$ and $\bar{D}$ represent total fixed and fission sources, the results of a source iteration are multiplied by the factor:

$$f^s = \frac{\bar{Q}^q}{\bar{Q}^q - (\bar{D}^q - \bar{D}^{q-1})} \quad (3.111)$$

An option is available to disable the procedure.

The results of an iteration, after rescaling and normalization if required, can be accelerated artificially by linear extrapolation. Where tilde indicates an unextrapolated value:

$$D_{i,j,s}^q = \tilde{D}_{i,j,s}^q + \theta(\tilde{D}_{i,j,s}^q - D_{i,j,s}^{q-1}) \quad (3.112)$$

The value of θ is adjusted such as to prevent a change of more than 20% in any D value. This procedure is sometimes helpful, but it has not been sufficiently reliable to be recommended. One weakness is the lack of a formal procedure to estimate updated fluxes consistent with the updated fission density.

Using the previously-described procedures, difficult source iteration problems may, nonetheless, fall into a pattern of slow convergence, with a slowly-diminishing change in fission density between each iteration. D. R. Vondy suggested the interpretation of this pattern as the decay of a single error mode,²⁴ which can often be used to extrapolate toward the correct result.

To accomplish this, DOT 4 accumulates the error sum e , the eigenvalue estimate λ , and an extrapolation parameter θ :

$$e^q = \sum \left| D_{i,j}^q - D_{i,j}^{q-1} \right| \quad (3.113)$$

$$\lambda^q = e^q / e^{q-1} \quad (3.114)$$

$$\theta^q = \lambda^q / (1 - \lambda^q) \quad (3.115)$$

When three successive estimates of θ^q agree within an input criterion, fission is extrapolated according to:

$$D_{i,j}^{q'} = D_{i,j}^q + \theta^q (D_{i,j}^q - D_{i,j}^{q-1}) \quad (3.116)$$

The value of q' is adjusted such that no fission density will have a fractional change greater than an input limit. Fluxes are similarly extrapolated.

Error-mode extrapolation can be used in conjunction with the other methods of acceleration described. It never causes harm, and is generally very effective in accelerating

K calculations, unless they are already converging rapidly. In the case of rapidly-convergent problems, a stable value of θ^p will not be established, and the extrapolation is not performed.

The extrapolation may also fail for fixed-source problems for which M is very high, due to computer roundoff. On an IBM 370, a problem with $M > 100$ may extrapolate at once to very near the correct result, and then not extrapolate again. If the flux shape due to fission is not vastly different from the shape due to the fixed source, the fission rescaling will reduce the effective value of θ^p , and extrapolation can be performed.

If the source is at the periphery of the system, however, fission rescaling may actually thwart convergence. For this reason, an option is available to defeat it.

3.16. Negative Source Repair

Even though non-negative flux models are used in serious problem solutions, false negative results may occur due to the finite expansion of the cross section data. For example, representation of photon scattering data in a P_l expansion with $l = 3$ produces troublesome negative and false positive scattering which may render the results useless. High-energy neutron scattering through roughly 90° or 180° is also likely to produce incorrect results. Analysis of the fits to the intended result in such cases has shown that the P_l procedure often results in good fits where the cross section is large, with meaningless oscillation between small positive and negative errors elsewhere. Rigorous repairs for this difficulty have been developed,³⁵ but none are in widespread use due to cost, complexity, or other limitations.

DOT 4 allows, as an option, a non-rigorous repair which requires no special data and may increase computing time by as little as 15%. As the source in each mesh interval is generated, all directions at that interval are examined for negative values. If negatives are found, the source in those directions is arbitrarily set to 0. In order to maintain balance, and to delete incorrect small positive values, the source in some of the remaining directions is set to 0, starting with the smallest and proceeding in order of ascending magnitude, until balance has been restored. If the negatives overwhelm the positives, the repair is abandoned.

This simple procedure was successful in solving a difficult skyshine problem,³⁶ and has met widespread application in radiation transport, and adverse effects upon results have not been observed.

3.17. Spatial Remeshing

In an earlier section, it was noted that various sets of X or R mesh boundaries could be used within the same problem. A brief description as to how the "remeshing" transition is done may be in order. The external boundary condition arrays are always supplied in a "standard I-set" mesh. A transition is made to the local mesh before beginning a sweep if the top row of the mesh does not contain the standard I-set. At each successive J-boundary, a remeshing is performed if required. At the bottom boundary, the result is remeshed into the standard I-set, if it is not already so represented, for determination of exiting flux arrays and for establishment of the boundary condition for the upward sweep.

The remeshing, itself, is performed on the flux along the entire row for each direction. For this purpose, the mean flux is assumed to apply across a given mesh interval. To calculate flux at interval i in mesh $R_1, R_2, \dots$ opposite a previously-determined flux in mesh $R'_1, R'_2, \dots$:

$$N_i = \frac{\sum_{i'} N_{i'} \left\{ \max \left[0, \min \left(R'_{i'+\frac{1}{2}}, R_{i+\frac{1}{2}} \right) - \max \left(R'_{i-\frac{1}{2}}, R_{i-\frac{1}{2}} \right) \right] \right\}}{R_{i+\frac{1}{2}} - R_{i-\frac{1}{2}}} \quad (3.117)$$

This procedure has only trivial cost, and can be very efficient when fine horizontal accuracy is required to detail a small feature. The user is cautioned, however, that the coarse mesh boundaries used in certain rebalance options must be contained in all I-sets.

Flux distortions can be produced by spatial remeshing, but they have been objectionable only when the mesh near the origin of an RZ or R θ problem is changed abruptly.

3.18. Directional Remeshing

As discussed earlier, direction-set (M-set) transitions can occur at I-boundaries, at J-boundaries, or between energy groups. The last case requires no special treatment, since the moment expansion of the flux is used to couple the groups. At I- or J-boundaries, "remeshing tables" are used to relate the old fluxes at each interval to the new fluxes. The tables consist of pairs of columns called "transition sets." A "transition table" locates the correct transition set to be used at a given transition. Clarification follows. Reference to a sample problem printout will help in following the discussion.

The transition table indicates which transition set governs the transition from the row M-Set to the column M-Set, e.g. a "1" appearing in row 1, column 2 means that transition set 1 governs the transition from M-Set 1 to M-Set 2. A negative indicates that the indicated table is to be used in the reverse direction. Thus, the transition table is always skew antisymmetric. A "0" entry means that a given transition does not appear in the problem, and no transition set will be generated for that transition.

The remesh table contains the actual transition sets. Each set consists of two columns of integers. Each position in each column corresponds to a direction in the corresponding M-Set, in "unpacked form." "Unpacked form" simply means that the tables have the length of the longest M-Sets, and "-1" has been inserted in certain unused positions as required by the code. Entries of -1 are to be ignored in the present procedure. The number of the transition set corresponding to the transition from M-Set n_1 to M-Set n_2 , may be found in transition table row n_1 , column n_2 . Let us call this $P_{2,1}$. The corresponding transition is then described in columns $2*|P_{2,1}|$ and $2*|P_{2,1}|-1$ of the remesh table. If $P_{2,1} > 0$, the lefthand column of the pair corresponds to M-Set n_1 ; otherwise it corresponds to n_2 .

The use of the columns is straightforward, if not simple. For example, to determine the value of flux in the direction m_1 in set n_1 , the code looks in the position m_1 of the column corresponding to n_1 which does not contain "-1." If an integer $m_2 > 0$ appears, the value of flux in direction m_1 , set n_1 , is set to the value of flux in direction m_2 in set n_2 . If

a "0" appears, the directions which have the integer m_1 stored in their corresponding positions in the M-Set n_2 are to be averaged appropriately to obtain the composite value of flux in direction m_1 . Since the tables are given in unpacked form, as needed by the code, the position in table n_2 corresponding to direction is the m_2^{-th} position which does not contain "-1."

Remeshing is performed over each octant of directions separately (e.g. for all directions with $\mu < 0$ and $\eta < 0$) at an interval, and then the fluxes in that octant are renormalized such as to preserve current across the boundary.

When several directions in M-Set n_2 are to be averaged to obtain a composite value of m_1 in M-Set n_1 , the procedure is as follows:

$$N_{m_1} = \frac{1}{W_{m_1}} \sum_m W_m N_m \quad ; \quad m \text{ in set } n_2 \quad (3.118)$$

The procedure for determining which type of transition is to be used is somewhat arbitrary, and not always entirely satisfactory. First, both columns of a transition pair (except -1 entries) are filled with the index of the "nearest neighbor" in the other M-Set, i.e., the direction which has maximum "directional closeness":

$$Z = \mu_{m_1} \mu_{m_2} + \eta_{m_1} \eta_{m_2} + \xi_{m_1} \xi_{m_2} \quad (3.119)$$

Then, the columns are examined for "subsector relationship." A subsector relationship exists if every direction in n_2 which was chosen as nearest neighbor by more than one direction in n_1 chose one of the directions which chose it, and if the same relationship holds true with the roles of n_2 and n_1 reversed.

In many transitions, such a relationship does exist, although the selections may be imperfect. If it does exist, each direction chosen by more than one direction in the alternate set has its transition integer set to 0, so that its flux will represent a composite of the corresponding fine-set fluxes. If not, each flux is simply set to the value of its nearest-neighbor flux. Other methods have been proposed, but none have survived preliminary evaluation. The subsector approach seems to have an important advantage in that each fine-mesh flux plays a role in determining a corresponding coarse-mesh flux, and no "uncoupled directions" which might affect stability exist.

Experience with the remeshing has been generally good. No pathological distortion or stability effects have been traced to the remeshing or to the use of multiple M-Sets. The cost of the procedure has proven negligible for practical problems.

3.19. Geometric Factors

Figure 3.2 shows the three geometries used in the discrete-ordinates solution. *The azimuthal angle is measured in units of rotations, rather than radians.* Each of the geometries has a "suppressed direction," i.e., Y in XZ geometry, θ in RZ geometry, and Z in R θ geometry. The area and volume elements are calculated assuming unity extent in the suppressed direction. In particular, an RZ geometric element extends for one full revolution. The elements A^V and A^H in the figure are A and B of Eq. 3.3.

ORNL-DWG E1-19464

● XZ (CUBE)

$$\begin{array}{lll}
 A^V = & A^H = & V = \\
 (\Delta Z) & \Delta X & \Delta X (\Delta Z)
 \end{array}$$

● RZ (PIE SLICE)

$$\begin{array}{lll}
 2\pi R (\Delta Z) & 2\pi \bar{R} \Delta R & 2\pi \bar{R} \Delta R (\Delta Z)
 \end{array}$$

● Rθ (CAKE SLICE)

$$\begin{array}{lll}
 2\pi R (\Delta \theta) & \Delta R & 2\pi \bar{R} \Delta R (\Delta \theta)
 \end{array}$$

Fig. 3.2. Geometric Elements.

Both A^V and V are two-dimensional in nature. For example, in RZ geometry, both depend upon R and ΔZ , and ΔZ depends upon Z . They are, however, factorable in a sense, and can be represented in the code by only 4 one-dimensional arrays, as shown in Table 3.2. In the table, arrays h and v represent storage areas into which the two sets of mesh boundaries are read, according to the geometry type. Arrays f^r , f^H , and f^z depend upon only variable h . $\bar{R}$ refers to a radius midway along the R interval. From these, the area and volume factors can be calculated as needed:

$$A^r = f^r \Delta v \quad (3.120)$$

$$A^H = f^H \Delta v \quad (3.121)$$

$$V = f^z \Delta v \quad (3.122)$$

3.20. Diffusion Theory

DOT 4 contains a diffusion-theory section borrowed from DOT III W,³⁴ by R. K. Disney. It was originally based on work by R. Protsik.³⁷ A special package developed for DOT 4 by D. B. Simpson¹ provides simulated boundary flows and associated data for use in balance table calculations. Certain discrete-ordinates options such as boundary sources, internal boundary flux output, etc. are not operable with the diffusion-theory section. The section is intended for use either to provide a good guess for a discrete-ordinates calculation or as a substitute for it.

Certain types of triangular geometry can also be solved by diffusion theory, as described in a later section. In general, more diffusion theory iterations are required as compared to discrete ordinates, but they execute at a speed approaching the speed of a 1-direction problem. Thus, a few of these iterations can provide an inexpensive startup for a discrete-ordinates K-calculation. This startup method is not particularly helpful for deep-penetration problems, however.

3.21. Units

DOT 4 can be used with either standard SI units or with the more traditional barn-centimeter-Angstrom units. Although we speak of the results as "fluxes" and "absorption rates," for example, fluences and total absorptions can be easily had by supplying source information which has been integrated over the desired time span. Table 3.3 illustrates both SI and traditional units.

Since the directional weights sum to unity, they are presumably unitless. Integrals over direction are replaced by weighted sums, and scalar fluxes, ϕ , are related to directional fluxes, ψ , as follows:

$$\phi = \sum W_m \psi_m \quad (3.123)$$

From this, it is evident that if ψ is uniform in m , $\phi = \psi$ and they must, therefore, have the same units. By dividing ψ by 4π , units of $1/m^2 \cdot s \cdot Sr$ can be obtained.

Also, by multiplying θ by 2π , units of radians (rad) can be obtained.

In XZ and R θ geometries, the "missing" direction is entirely ignored. Thus, the results are normalized as if the system were of unity size in Y and Z, respectively.

Table 3.2. "Factored" Geometric Arrays

Geometry	XZ	RZ	R θ
Input Array h	X	R	R
Input Array v	Z	Z	θ
Calculated Array Δv	ΔZ	ΔZ	$\Delta\theta$
Calculated Array f^v	1	$2\pi h$	$2\pi h$
Calculated Array f^H	Δh	$2\pi h \Delta h$	Δh
Calculated Array f^S	Δh	$2\pi h \Delta h$	$2\pi h \Delta h$

Table 3.3. DOT 4 Unit Systems

	SI Units	Optional Traditional Units
Space Dimensions, R, Z, X :	Meters (m)	Centimeters (cm)
Time, t :	Seconds(s)	
Rotation Fraction, θ :	Dimensionless	
Weight, W :	Dimensionless	
Atomic Densities :	$1/m^3$	$1/A^3$ (Atoms per 10^{-24} cm^3)
Micro Cross Sections, σ :	m^2	barn (10^{-24} cm^2)
Macro Cross Sections, σ :	$1/m$	$1/\text{cm}$
Scalar Fluxes, $\bar{N}$:	$1/m^2 \cdot s$	$1/\text{cm}^2 \cdot s$
Directional Fluxes, N :	$1/m^2 \cdot s$	$1/\text{cm}^2 \cdot s$
Boundary Sources :	Same as N	Same as N
Distributed Sources :	$1/m^3 \cdot s$	$1/\text{cm}^3 \cdot s$

Region 4. Programmer's Information

4.1. Code Structure

The code is arranged in four essentially independent functional units, linked only by data contained in the common block and in the external data files. These units, together with the "control subprogram" applicable, are:

1. Parameter Input (LOCO) - input and edit of parameters controlling problem size and logic path.
2. Array Input (INPUT) - input of primary dimension-setting arrays describing space mesh, variation of directional quadrature over the mesh, and extent of moment expansion; input of secondary dimension setting arrays describing the direction sets and various coarse meshes; input of general arrays describing other problem conditions; and input of optional arrays for internal generation of flux and source data input files.
3. Computation (WORK) - solution of the transport problem and gathering of certain data to be edited. Certain output data files are generated here if necessary.
4. Output (OUTPUT) - final normalizing and editing of results. Writes output data files not processed by WORK.

4.2. Inter-machine Adaptability

The code is intended to be easily adaptable to any type of sophisticated computer, and yet to take advantage of certain localized structural features which may be machine-dependent.

In general, the guidelines of ANS-STD.3-1971³⁸ are followed. This requires general adherence to a simple, standard FORTRAN language except where deviations provide important improvements in capability and can be documented. In addition, machine-dependent features have been kept localized and have been kept away from the subroutines which perform actual computation.

The recommended procedures of the Committee on Computer Code Coordination (CCCC)^{39,40} have been followed where practical. Compliance with these standards has been incomplete where cost was prohibitive, the nature of the code did not allow it, or the expected level of performance could not be obtained in that fashion.

Where minor machine-dependent features are required, alternate features are enclosed in pairs of 3-character "language flags." The alternative statements remain in the source program, with inappropriate sections transformed into comment statements. If the code is set up for IBM operation, for example, it might contain:

```

CIB
      ENTRY IBCDC(H,E,L,P)
CIB
CDC
C      ENTRY IBCDC
CDC

```

The corresponding CDC configuration would be:

```

CIB
    ENTRY IBCDC(H,E,L,P)
CIB
CDC
C    ENTRY IBCDC
CDC

```

All versions of a given procedure are thus available for inspection by all users. The selection of options is made by a computer program at distribution time. A listing of machine-dependent sections can also be prepared in that process.

The language flags achieve the major objective of having a single, unified source program which is maintained for all users. With these provisions, the basic FORTRAN program is operable on IBM, CDC, UNIVAC, CRAY, Amdahl, and other computers, except that system-dependent service routines as specified by CCCC must be provided.

Optional packages and procedures which provide enhanced performance and convenience are also available. The optional run-time storage allocation described later requires system-dependent routines. On IBM systems, three optional assembler-language subroutines are also supplied to increase the efficiency of execution. These approximately double the execution speed on pipeline computers. They are also valuable to sequential computers such as the IBM 370/3033, but to a lesser extent. All known users at this time use all of the optional enhancements available to them. Even so, the basic operation with FORTRAN routines plus the CCCC package remains available if needed.

4.3. Service Subroutines

Certain standard service subroutines are specified by the CCCC for use in reactor physics codes. The service routines used in this code include:

TIMER	Provides timing and job identification data
REED	} Provide sequential access to data files
RITE	
DRED	} Provide random access to data files
DRIT	
DOPC	Provides initiating, closing, and certain repositioning of data files
CRED	} Provide block transfer of data between fast and slow memory.
CRIT	

Many of these subroutines also call other subroutines. Many are entirely system dependent. Each configuration to be distributed contains an appropriate set of service subroutines, insofar as possible. The realities of computing environments may require local modification or substitution. The specifications given in Reference 40, together with extensive in-stream comments, provide guides for such modification.

4.4. Programming Style and Practices

The programming style is greatly affected by the irregular mesh requirements. As an example, the simple task of finding the zone number for space cell (I,J) is normally:

```
DIMENSION IJZN(IM,JM)
      K = IJZN(I,J)
```

If IM depends upon J, however, this is not adequate. Instead, we assemble "indexing pivot" arrays:

$$IBJ(J) = \sum_{I=0}^{J-1} IM(I) \quad ; \quad IM(0) = 0$$

whereupon, in standard FORTRAN:

```
CALL SUBA(D(LIJZN),D(LIBJ),JM)
```

```

SUBROUTINE SUBA(IJZN,IBJ,JM)
  DIMENSION IJZN(1),IBJ(1)
  IJ=1+IBJ(J)
  K=IJZN(IJ)
  RETURN
END
```

Although cumbersome, this optimizes into code nearly as efficient as normal FORTRAN. Most of the data arrays are singly dimensioned because of the irregular mesh features.

A set of "scratch parameters," I1 through I20 and E1 through E9, are located in CMDOT for general use. These provide some space savings and help to distinguish parameters whose use is only over the span of a few statements from parameters of real significance, to which mnemonic names are attached.

Although special language requirements have been carefully avoided, a few practices which are required by the nature of the code deserve mention. None of these practices have been troublesome as yet.

It is essential to operation of the code that unneeded dimensions, i.e., the length of singly dimensioned arrays and the last dimension of multiply dimensioned arrays, can be set to 1 if the arrays are only pseudonyms for space in the container array and if they are not used in READ or WRITE statements. Without this feature, the argument lists would be too long for some computers.

It is essential that a given area of the container can be referred to as either real or integer in various places, so long as the type is consistent with the data being stored; e.g., in the example above, the real array D contains integer data array IJZN. Also, D(IJZN) and IJZN(1) refer to the same locations in subroutines which contain both D and IJZN. This is unavoidable, and causes no real difficulty. It does, however, require that real and integer words have the same length.

In FIDAS, for example, the same array may be given two or more names in the argument list and either name may be used to update the information, according to type. Non-subscripted data items are never treated in this manner, however. Warning comments usually flag such occurrences, except in the case of the container array.

In a few areas, real-type statements such as $A(I)=B(I)$ may be used to move integer data, or vice versa. But $A(I)=-B(I)$ is never used in such a case.

4.5. Overlay Arrangements

Two overlay arrangements are available with the code. The first, called the "segment structure," is applicable to IBM and UNIVAC systems or to CDC systems which support segmentation. This structure is shown in Fig. 4.1. In addition to the items shown, common blocks, together with a dummy subroutine HEADER, may be located by means of segmentation directives on certain systems.

The second arrangement is intended to allow operation on a CDC system which supports only the CALL OVERLAY statement. In this system, each program unit begins with a main program which has no arguments. The appropriate subprogram grouping is shown in Fig. 4.2. Although the OVERLAY feature is helpful in increasing the amount of memory available for data, it is not as effective as the segmentation loader. In addition, it requires that a special source program be prepared by appropriate changes to the language flags.

The segmentation arrangements are in daily use at various facilities, and should be almost trouble-free. The CDC OVERLAY facilities may not function properly as distributed, due to disuse, and may require local modification.

4.6. Internal Storage Management

The code uses "flexible dimensioning," in which all data arrays of non-trivial length are assigned dimensions at execution time and are kept in a single large "container array." The beginning of each array is indicated by an "array locator" corresponding to that array. Array locators are kept together in a special labeled common block. For example, where D is the FORTRAN name of the container array, LCHI would be the array locator corresponding to array CHI. Once the value of LCHI has been set, the CHI array is made available to a subroutine by, for example:

Fig. 4.1 Segmentation Structure.

OVERLAY (0,0)

 Contents of Root Segment in Previous Figure

OVERLAY (1,0) Contents of Segment B1A		OVERLAY (2,0) Contents of Segments B1C C2A C2B C2C D2A D2B D2C	OVERLAY (3,0) Contents of Segment B1B
OVERLAY (1,1) Contents of Segment C1A	OVERLAY (1,2) Contents of Segment C1B		

Fig. 4.2. Overlay Structure.
 (Refer to previous figure for contents of segments)

```
CALL SUB(D(LCHI),...)
```

```
...
```

```
SUBROUTINE SUB(CHI,...)
  DIMENSION CHI(1),...
  X=CHI(N)
```

```
...
```

In general, the last dimension of each array is set to unity inside the called routine. Failure to do this causes trouble on computers which have a limited argument-list capacity.

In the basic "all-FORTRAN" operation, the code keeps the container array in a labeled common block, CMBLK. Its length is declared in a standard dimension statement, and arrays are packed into that space as required. Additional details of the array requirements are given in a later section.

To facilitate adaptation to various computers, CMBLK appears in only an essential few subroutines. It is passed to the other subroutines through the argument list as required. Allocation of space in the container is actually controlled by subroutine ALOCAF. ALOCAF is given the maximum allowable container length at compile time. It responds to requests for space by supplying an initial location to be used by the requesting routine, and by tallying a total of all requests. On UNIVAC, CDC, and CRAY computers, the segmentation loader is used in a way such as to load the container array at the end of the program space.

An optional capability for run-time storage allocation is also provided. This provides so much additional convenience and economy that virtually all users maintain the code in this configuration if possible. Suitable configurations for UNIVAC, CDC, CRAY, and IBM are available, and distributions prepared for these computers give instructions for installing this option.

Several labeled common blocks are used to contain individual data items and a few small arrays:

CMLOC	The locators specifying array origins in the container array.
CMDOT	Input and code-generated problem parameters of general significance.
CMFLX	Parameters used primarily during the flux calculation.
IOREC	Data set status information.
COMIN	Standard input-output devices, error processing, titles, word length.
[COMSAM]	Status information for the random access routines. These are used only by certain CCCC routines and routines which affect them.
[COMCDC]	
COMROW	Parameters used by ROW and its subordinate subroutines.
CMBLK	Container array.

On UNIVAC and IBM systems, a certain amount of space must be reserved for the I/O routines. This is called "system buffer" space although it also contains certain blocks of control data required for I/O. An estimate of the space required for this function is given in the input data, and the required amount of space is reserved from the pool available to ALOCAF for this purpose. Additional details are given in a later section.

4.7. Split-Memory Features

A "split-memory" configuration option is available for CDC 7600 and CYBER 176 users. It uses the slow memory on these computers primarily for storage of large data arrays which are to be accessed by block-move routines and by I/O routines. In addition, a few directly-addressed arrays are assigned to the slow memory. By so assigning only large, infrequently-used arrays, no measurable reduction in execution speed due to the use of slow memory has been experienced. The older option remains in the source deck, although it is no longer used or checked.

The directly-addressed arrays in slow memory are declared in LEVEL 2 statements wherever they are used. They are subject to an overall hardware limitation of 131,000. The data accessed by the block-move and I/O routines are not subject to this limitation.

4.8. External Storage Management

Data are stored externally on large-scale storage devices in "data sets" or files." Each file is assigned a "logical unit number" (LUN) less than 100, and is accessed by a "logical name" in the program into which is set the logical unit number. As a matter of convenience, user-supplied input files and results files may be spoken of as "tapes," while scratch files may be spoken of as "disks," although other equivalent storage can be used.

All data communication with input and result files is sequential in nature, managed by the SEQIO routine, and accomplished by REED and RITE. The versions of these subroutines supplied with the code should be directly applicable to all systems. Copious comment cards identify the function of these and other data-handling routines.

Scratch file data communication is channeled through VARIO, and falls into three classes. The "sequential" files, LUN's 81-89, can be processed adequately by the sequential I/O package, SEQIO, REED, and RITE, and this is done on CDC systems. On UNIVAC, IBM, and CRAY systems, a performance advantage is gained by routing such data through BLKIO, DOPC, DRED, and DRIT.

The "quasi-sequential" class could be processed adequately by SEQIO if it were possible to rewrite a record without destroying subsequent records. The boundary directional flux file is an example of such application. At the present time, LUN's 94 and 95 are quasi-sequential, and are processed by the random-access I/O package, BLKIO, DOPC, DRED, and DRIT.

The third class of communication is "blocked-random" access, used for flux and source moment files, 91-93. Records in this class are written sequentially, but may be read or rewritten in any order. The information is brought into "users buffers" before use, and may be used or updated at random before it is returned to the external file. Additional details are to follow.

Unit assignments on UNIVAC are different from those of other systems due to a limitation of 29 on LUN.

4.9. Moment File Management

With the exception of flux and source moments, working data are assigned to fixed memory arrays or scratch files. The size of the moment files, however, requires more flexibility. A state-of-the-art problem may have 300,000 words of flux moment data, and twice that amount in source data, for only one energy group. With 100 energy groups, a flux file of 30 million words would be required, and the long-term trend in problem size is upward. A partial solution is to store the moments on a scratch file, to be used a group at a time and then returned. The DOT III code used this operating mode. This mode may severely limit problem size, however. Consider, for example, a CDC computer with 130,000 words devoted to directly-addressed slow-memory arrays and 200,000 words available for moment storage. Assuming P_3 scattering, mesh size would be limited to less than 7,000 cells. Similar considerations are applicable to other systems.

The multilevel storage concept allows considerably more flexibility without impairment of execution speed. It is permitted by CCCC standards.⁴⁰ In the DOT application, two distinct storage modes are used. In the first, "problem stored," mode, all flux and source moments are stored in contiguous arrays in memory and addressed in the normal fashion. For smaller problems, especially on some virtual systems, this mode brings top efficiency.

If the available memory is not sufficient for the problem-stored mode, the "row-stored" mode, illustrated in Fig. 4.3, is used. A row is simply all of the space-mesh cells having a common second-index value, i.e., all cells for a given j -level. Data for a single row are kept in the "working memory," and moved to and from the "user buffer" as required by high-efficiency block-move routines CRED and CRIT. The user buffer may be read or written at random as required. Its information may be moved to a scratch file when another block of data is required.

The computation section, then, sees only two storage modes, either problem or row mode. The complexity required to manipulate various buffer and scratch-file arrangements is contained in VARIO and its associated data management routines.

An important advantage is obtained when the user buffers are large enough to contain all of the data for an energy group ("group-stored"). In this case, the solution of an entire energy group can be accomplished without intervening I/O activity. I/O of flux moments takes place only before and after the calculation of flux results for a group. In such a case, scratch files for source moments are not required at all.

The impact on both CPU time and CPU utilization (ratio of CPU time used to real time used) of changing from problem-stored mode to group-stored mode is small. The increase in CPU time is usually barely measurable. CPU utilization tends to be in the 60% to 90% range in either problem-stored or group-stored mode, unless other users compete for system resources successfully. With row-stored mode, utilization may drop to the 30% to 60% range, and it may be important to run concurrently with other problems which can share the CPU.

Fig. 4.3. Split-Memory Storage Example.

Region 5. User's Information

5.1. Input Data Specifications

5.1.1. Card-Image Format

The card-image input data for a problem consist of a title card followed by a variable number of blocks of data separated by "T" delimiters. Within each block, a variable number of data arrays are specified, keyed according to an array number, and identified as to real or integer type.

A separator card must be placed between successive problem decks. If it contains the word "DIAG" in the first four columns, a non-fatal diagnostic will be given by ERRO. If the word "DUMP" appears, a fatal diagnostic is given. If the word "=END" appears, execution will stop. Otherwise, the next problem begins immediately. No data are retained in memory between problems. At the end of execution, all data files are closed. Control is then returned to the driver, if the code was invoked by a driver, or else to the operating system.

All of the card-image input except the title card and separator card are read by the FIDO input processor. The format required for FIDO data entries is described in Appendix A. The data arrays and blocks to be read are described in following sections. The first four blocks of data are permanent, and their "T" delimiter must be supplied whether array entries are required or not.

Blocks, after the first four, containing arrays numbered 91 and higher, are options supplied to allow certain simple problems to be solved without the labor of preparing formal flux and source input files. The conditions under which these blocks are required are included in their description.

As each array of data is read, its length is compared with the required length, and an error message is given if appropriate. After each block of data, a call to subroutine ERRO is made if any of the arrays in the block were of improper length, or if the data were otherwise unsuitable. ERRO prints an explanation of the error and sets an error flag. Data are edited as soon as possible after each block is completed.

Processing of input continues for as long as possible, even after an error. In most cases, this allows the user to test all of his data blocks in spite of an early error. Execution is terminated at the end of input processing if a sufficiently severe error has been encountered.

It is possible that one error will cause subsequent data to be erroneously flagged as wrong. If data which appear correct are flagged by an error message, this possibility should be considered. An error can also cause subsequent system related failures, such as addressing or storage errors.

The first four blocks of data contain, in general terms:

1. control parameter input arrays (61-63) which set array lengths explicitly and control execution options.

2. primary input arrays (71-78) whose lengths depend upon 61-63, and whose contents determine later array lengths. These arrays describe variable-mesh, variable quadrature, variable P_L , and super-mesh features.
3. secondary input arrays (81-87), whose length depend upon 61-78, and whose contents determine later array lengths. These include the directional quadrature specification set, coarse mesh specification, assignment of direction sets to super-mesh cells, and assignment of material zones to edit regions.
4. general input arrays (1-30), which do not determine the lengths of other input arrays. These include space-mesh specifications, cross section mixing instructions, specification of activity edits, and much more.

In the remainder of Region 5 of this report, the broad concepts required to understand the input data are discussed, followed by a detailed specification of the input data, and finally, clarification of several input-related topics.

5.1.2. Space Meshes

Five different subdivisions of the problem space are determined from the input data. Two sets of fine-mesh interval boundaries (2^* and 4^* arrays) are input, corresponding to the two spatial dimensions. The intersection of these boundaries forms the fine-mesh grid cells. (In $R\theta$ geometry, the 2^* array is entered in units of revolutions, not radians.)

A coarse mesh, specified by an additional 2 sets of intersecting boundaries (85^* and 86^* arrays), is used in the flux acceleration routines. A good choice of coarse mesh can save both memory space and execution time without impairing the convergence of a problem. In some cases, proper application of the coarse mesh can help the convergence rate as well. A coarse mesh of one mean-free-path is ideal for many problems. Default options allow the coarse mesh to be identical to the fine mesh if the variable space mesh feature (described later) is not used. A coarse-mesh I-boundary must also be a boundary in each of the fine-mesh I-boundary sets. A coarse-mesh J-boundary must also be a fine-mesh J-boundary.

A super mesh is similarly specified (75^* and 76^* arrays) for use with the variable quadrature feature (described later), and with certain search options. A super-group representation ($74\$$ array) is also required when variable quadrature is used. The location of the super-mesh and super-group boundaries is chosen by the user as desired, except that the super-mesh boundaries must also be fine-mesh boundaries.

Material zones ($8\$$ array) are always specified. A material zone must contain only one cross section set, although many zones may share a single cross section set. In addition to cross section assignment, certain input and edit features depend upon material zones.

Edit regions ($84\$$ array) are groupings of material zones used for the purpose of condensing certain output tables. Certain input and edit features depend upon edit regions.

5.1.3. Variable Space Mesh

The fine mesh can be specified such that the I mesh depends upon J. To accomplish this, several "I-sets" are specified, with different numbers of intervals and different interval boundaries. The ISET(J) array then relates the proper set of boundaries to the J-level.

Prudent use of this feature can save computation time and storage by concentrating work in areas requiring a fine mesh. I-direction coarse-mesh and super-mesh boundaries must be contained in all fine-mesh I-sets. Left and right boundaries of all I-sets must coincide. A "standard" I-set, which is as large as any of the I-sets, must be chosen. Boundary fluxes and sources are kept in the standard I-set format.

5.1.4. Variable Directional Quadrature

The directional quadrature sets, "M-sets," are specified by super-zone and super-group. For example, this allows the user to specify a highly biased quadrature in the area of a streaming crack at high energy without wasting such detail elsewhere in the problem. A standard M-set must be chosen which:

1. has as many directions up and as many down as any set,
2. has exactly $1MM$ directions, and
3. has as many η levels upward and as many downward as any set.

5.1.5. Variable Legendre Expansion

The order of moment expansion (P_L) may be specified by cross section set (78\$) and by energy group (77\$). The former option allows selective examination of anisotropic effects, while the group dependence can save computation time in lower energy groups, where anisotropy is less pronounced.

(NOTE: These features have not been made operable in current versions of the code.)

5.1.6. Adjoint Data

If an adjoint problem is to be solved, all data in the first four blocks are entered in the normal manner. All input files supplied by the user, and any arrays numbered 91 and higher, must be supplied in adjoint form, i.e., reversed with respect to energy. For files having directional information, the user should remember that an adjoint problem is solved as a function of $-\underline{\Omega}$, rather than $\underline{\Omega}$, where $\underline{\Omega}$ is a direction specified by the quadrature set. In other words, if $\mu > 0$ and $\eta > 0$ for a given direction, the adjoint data calculated for that direction will be that appropriate to a particle moving in the direction $(-\mu, -\eta)$. Directional input files must be similarly reversed, and directional output files will be. Output files have the energy groups in the order calculated, i.e., reversed from the usual order.

5.1.7. Card-Image Input Specifications

In the following section, the symbol "@" indicates that a detailed discussion is given elsewhere in this region. The symbol "#" denotes the word "number," and "RV" means "recommended value." After each array, the length is specified in parentheses. If an array or block is not always supplied, the conditional requirement is given in brackets. The symbol "*" denotes a feature not presently operable.

A. Title Card (72 alphameric characters).

B. Control Input Arrays

61\$ Data Set Logical Unit Reference Numbers@
(Leaving a unit as 0 disables the corresponding feature;
the length of this array is subject to change.)

	NTFLX	flux guess input unit
	NTFOG	flux output unit
	NTSIG	cross-section unit (always required) (default=8)
	NTBSI	external boundary source input
(5)	NTDSI	distributed source input unit (scratch unit must be supplied if INPSRM.GT.0)
	NTFCI	first-collision source input unit
	NTIBI	internal boundary source input unit
	NTIBO	internal boundary flux output unit
	NTNPR	large-scale print unit (all print goes on standard output unit if NTNPR = 0)
(10)	NTDIR	directional flux output unit
	NTDSO	distributed source output unit

(terminate array with "E")

62\$ Integer Control Parameters
(length subject to change)

	IADJ	0/1 = forward/adjoint calculation
	ISCTM	maximum order of scattering (L of P_L) (negative indicates P_L calculation is specified by energy group (77\$ array))*
	IZM	number of material zones
	IM	maximum number of 1st-dimensional (I) spatial intervals in any i-set (negative indicates number of intervals varies with 2nd dimension)
(5)	JM	number of 2nd-dimension (J) spatial intervals
	IGM	number of energy groups
	IHT	position of total cross section in cross-section table@
	IHS	position of self-scatter cross sections in cross-section table @
	IHM	length of cross-section table for each group@
(10)	MIXL	mixing table length@
	MCR	maximum order of Legendre expansion of cross sections (negative indicates expansion is specified by cross section set (78\$ array))*
	MTP	number of cross-section sets from NTSIG (0 has the effect of MTP=MTM)@
	MTM	total number of materials, including mixtures@

	IDFAC	density factor input option 0 feature not used 1 DNIJ(IJ) input as 3*, used to modify cross sections
(15)	MM	maximum number of directions in any M-set (negative indicates multiple M-Sets are used)
	INGEOM	geometry option@ 0 X-Z slab 1 R-Z cylinder 2 R- θ circle 3 180° - 360° triangular 4 60° triangular 5 90° triangular * 6 120° triangular
	IBL	left boundary condition 0 void 1 reflected 2 periodic 3 cylindrical 4 fixed boundary source 5 albedo
	IBR	right boundary condition 0/1/2/3/4/5 (see IBL)
	IBB	bottom boundary condition 0/1/2/3/4/5 (see IBL)
(20)	IBT	top boundary condition 0/1/2/3/4/5 (see IBL)
	ISRMX	outer iteration maximum
	IFXMI	initial inner iteration maximum per group@ (negative indicates maximum given by group (28\$ array))
	IFXMF	final inner iteration maximum per group@ (no effect if 0)
	MODE	flux extrapolation model 0 linear with negatives set to 0 1 linear with no fixup of negatives 2 step 3 weighted difference 5 θ -weighted
(25)	KTYPE	calculation type@ 0 fixed source 1 K eigenvalue 2 DB ² search 3 concentration search 4 dimension search
	IACC	rebalance method in discrete ordinates section (default = 2)@ 0 single groupwise rebalance factor 1 diffusion acceleration 2 partial current rebalance
	KALF	rebalance stabilization@ 0 standard method 1 alternate method

	IGTYPE	solution method@ 0 discrete ordinates used always N method selected by group (7\$ array) for first N outers, finishing with discrete ordinates
	INPFXM	flux input option 0 flux set to 0, or read from unit NTFIX if NTFIX>0 1 FIJ(IJ) read as 93* array for each group 2 FIJ(IJ)*FG(G), uses 93* and 95* arrays 3 FI(I)*FJ(J)*FG(G), uses 93*, 94*, and 95* arrays
(30)	INPSRM	distributed source input options 0/1/2/3 (options are analogous to INPFXM; use 96*, 97*, 98* arrays) (NTDSI>0 is required if INPSRM>0)
	NJNTSR	interior boundary source at NJNTSR J-boundaries will be input from NTIBI (may be 0)
	NINTSR	interior boundary source at NINTSR I-boundaries will be input from NTIBI (may be 0)
	NJNTFX	interior boundary flux at NJNTFX J-boundaries will be written on NTIBO (may be 0)
	NINTFX	interior boundary flux at NINTFX I-boundaries will be written on NTIBO (may be 0)
(35)	IACT	number of region and pointwise activities calculated@ (negative indicates region activities only)
	IREC	balance table output control In general, IREC is the largest number in the 84\$ array, except that: IREC=IZM one region for each zone, balances for IZM regions IREC=1 combines all zones into a single region IREC=0 no balance table output, but one region for each zone IREC=-1 provides one region per zone, balance tables will be given for the entire system only
	IPDB2	punch J-direction DB ² from balance table output 0 no DB ² produced 1 single average DB ² 2 IGM groupwise DB ² values 3 IGM*IREC groupwise and regionwise DB ² values (use negative entry to bypass balance table print)
	IFXPRT	scalar flux print option (default = 1) 0 scalar flux for all groups printed in output phase 1 scalar flux not printed 2 scalar flux printed as calculated
	ICSPRT	cross section print option (default = 1) 0 cross sections printed 1 cross sections not printed
(40)	IDIRF	directional flux 0 directional flux not saved 1 directional flux saved and printed 2 directional flux saved but not printed (directional flux will be written onto NTDIR if IDIRF>0 and NTDIR>0)

	JDIRF	first J-interval for directional flux output
	JDIRL	last J-interval for directional flux output
	NBUF	system buffer space@ IBM: K-byte units, default=60 UNIVAC: K-word units, default=2 CDC: no effect
	IEPSBZ	zone importance convergence (uses 24* array)@ 0 feature not used 1 zone importance used 11 as 1 + zone convergence printed after convergence 21 as 1 + zone convergence printed after each iteration
(45)	MINBLK	minimum J-blocking for flux moment storage@ 0 1 space block per group, all groups in memory allowed 1 1 space block per group, 1 group in memory allowed N at least N space blocks required, 1 group in memory
	MAXBLK	maximum J-blocking@ 0 JM space blocks per group allowed 1 only 1 space block per group allowed N maximum of N space blocks per group allowed
	ISBT	standard I-set for boundary fluxes (default=1)
	MSBT	standard M-set for boundary fluxes (default=1)
	MSDM	standard M-set for dimensioning (default=1) (MSDM=MSBT always in this version)
(50)	IBFSCL	number of flux iterations before first rebalance in first source iteration@ (default=1)
	INTSCL	minimum number of rebalance iterations (default=4)@
	ITMSCL	maximum number of rebalance iterations (default=100)@
	NOFIS	fission spectrum option 0 fission calculated with χ normalized to 1.0 1 fission calculated with input χ values 2 fission calculation bypassed, saving memory space (NOFIS will be set to 2 if the 1* array, chi, is 0)
	IFDB2Z	DB ² input option 0 option not used 1 DB2Z(IG,IZ) entered as 6* array
(55)	ISWP	type of diffusion theory sweep 0 line inversion in direction of largest number of mesh intervals 1 alternating direction (line-column) on consecutive outer iterations 2 line inversion 3 column inversion 4 alternating direction (line-column) on consecutive inner iterations 5 alternating direction (column-line) on consecutive inner iterations 6 line inversion outward from center J interval
	KEYJN	J-interval for key flux print. (ignored if 0)@

	KEYIN	I-interval for key flux print (ignored if 0)@
	NSIGTP	cross section input file (NTSIG) format@ 0 GIP format 1 ORDOSW format
	NORPOS	cross section table position for output normalization (option not used if NORPOS=0)
(60)	NORMAT	material number for output normalization (0 implies use macro cross section set; negative implies do not use density factor in normalization)
	MSTMAX	maximum number of M-sets (0 implies JM sets allowed)
	NEGFIX	negative scattering source fixup@ 0 option not used 1 repair used
	LOCOBJ	fast memory objective into which problem will try to fit@ 0 implies: CDC 38 K-words UNIVAC 25 K-words IBM no effect
	LCMOBJ	slow memory objective into which problem will try to fit@ 0 implies: CDC 131 K-words others no effect
(65)	NKEYFX	length of key flux arrays (29\$, 30\$)@ (0 bypasses feature, negative causes print only at convergence or iteration limit for each group)
	NCNDIN	maximum user condition code allowed (default=1)
	NEUT	last neutron group number (0 implies NEUT=1GM)

(terminate array with "E")

63*

real control parameters
(length subject to change)

	TMAX	maximum minutes of CPU time for this problem (0 ignored)@
	XNF	value to which source is normalized (0 ignored if KTYPE=0, otherwise 0 implies XNF=1.0)
	EPS	eigenvalue convergence criterion (source iterations) (default=1.0E-4)@
	EPP	pointwise flux convergence criterion (flux iterations) (default=1.0E-3)@
(5)	EPV	volumetric flux convergence criterion (flux iterations)@
	EPF	Pointwise fission convergence criterion (default=1.0E-3)@
	EKOBJ	k-effective sought in search (KTYPE>1) or initial k-effective (KTYPE=1) (default=1.0)@
	EVTH	k-effective convergence ratio (default=0.2)@
(10)	EVCHM	maximum EV change ratio per iteration (default=1.5)@
	EVMAX	maximum EV change ratio, overall (default=10.0)@
	EVKMX	maximum allowed K-effective -EKOBJ (default=1.0)@
	EVI	initial eigenvalue (default=1.0)@
	DEVDKI	initial eigenvalue slope (default=-1.0)@
	EVDELK	initial eigenvalue increment (default=-0.3)@

(15)	SORMIN	maximum error-mode adjustment ratio (default=1.2)@
	CONACC	rebalance acceptance criterion (default=3.0E-2)@
	CONSCL	spatial rebalance convergence criterion (default=1.0E-4)@
	CONFIX	reserved (use 0)
	WSOLOI	rebalance minimum scattering ratio (RV=0.3)
(20)	WSOLII	rebalance flux iteration increment (default=0.5) (negative indicates use IWSOLIII only as required)
	WSOLCN	rebalance constant (default=1.0)
	ORF	diffusion theory flux acceleration factor (default=0.6)
	FSNACC	fission density acceleration factor (RV=0)@
	FLXMIN	minimum flux for convergence tests@ (default: UNIVAC 1.0E-30 others 1.0E-60)
(25)	SMOOTH	reserved (use 0)
	EPO	pointwise flux convergence criterion (source iterations)@
	EXTRCV	source iteration extrapolation convergence criterion@ (default=0.1)
	THETA	theta-weighted extrapolation model parameter (default=0.9)

(terminate array with "E")

(terminate block with "T")

C. Primary Dimension-Setting Arrays (Omit arrays not needed).

71\$	ISSET(J) (#=JM) [IM<0]	Index of radial mesh set to use at each J level. Default = Set 1
72\$	IMBIS(ISET) (#=JM) [IM<0]	Number of intervals in each I set; then fill array with 0's.
73\$	MMBMS(MSET) (#=MSTMAX) [MM<0]	Number of directions in each M set; then fill array with 0's.
74\$	ISZNG(IG) (#=IGM) [MM<0]	Super group number by group Default = a single supergroup
75*	SZNBZ(JSZ) (#=JM) [MM<0 or KTYPE>1]	J super mesh boundaries Enter as many boundaries as desired. Then fill with 0. Default = a single superzone
76*	SZNBR(ISZ) (#=IIM) [MM<0 or KTYPE>1]	I super mesh boundaries Enter as in 75*

77\$ ISCTG(IG) (#=IGM) [ISCTM<0]
Order of Legendre expansion used in calculation *

78\$ NSIG(MT) (#=MTM) [MCR<0]
Order of Legendre expansion of cross section sets *

(terminate block with "T") [T always required]

From these, the following are determined:

MGSZN	= largest super group number [1 if MM>0]
NJSZN	= number of J super zone boundaries [1 if MM>0]
NISZN	= number of I super zone boundaries [1 if MM>0]
ISM	= # non-zero entries in 72\$ [1 if IM>0]
MSM	= # non-zero entries in 73\$ [1 if MM>0]
IMSISM	= sum of IMBIS, all I sets [IM if IM>0]
MMSMSM	= sum of MMBMS, all M sets [MM if MM>0]
IMSJM	= sum of IMPIS(ISET(J)), all J [IM*JM if IM>0]
IMA	= IM
MMA	= MM
MMSIMS	= MMA*IMA
IMSJM	= MMA*JM
IHP	= IHM + 1 if IHS>IHT + 1; otherwise = IHM

D. Secondary Dimension-Setting Arrays (omit arrays not needed)

81* W(M,MSET) (#=MMSMSM)
Directional weights

82* EMU(M,MSET) (#=MMSMSM)
 μ , cosine of angle with X or R direction

83* ETA(M,MSET) (#=MMSMSM)
 τ , cosine of angle with Z or θ direction

84\$ IZNRG(IZ) (#=IZM) [IRED $\neq$ 0 or IACT $\neq$ 0]
Region number by zone (default controlled by value of IRED,
one region per zone if IRED=IZM, 0, or -1; one region if
IZM=1; otherwise IZNRG must be specified in full)

85* ZCMB(JC) (#=JM)
J coarse mesh boundaries
Enter as many boundaries as desired up to the limit, JM.
Then fill with any number at least as large as the last J
fine-mesh boundary.
(Default = 1 course mesh for each interval)

86* RCMB(IC) (#=iMA)
I coarse mesh boundaries, as in 85*

87\$ IJGSZ(ISZ,JSZ,IGSZ) (#=NISZN*NJSZN*NGSZN) [MM<0]
 M set by I super mesh, then by J
 super mesh, then by super group

(terminate block with "T")

From these, the following are determined:

JCM = # of coarse mesh boundaries entered in the ZCMB array
 ICM = # of coarse mesh boundaries entered in the RCMB array
 NREG = maximum of IZNRG(IZ) array
 ICMJCM = ICM*JCM
 ICPJCP = (ICM+1)*(JCM+1)

E. Data Arrays (Omit arrays not needed)

1* CHI(IG) fission spectrum fractions, χ , by group (#=IGM)
 2* ZIN(J) Z or θ fine-mesh boundaries (#=JM+1)
 3* DNIJ(I,J) density factor (#=IMSJM) [IDFAC>0]
 (Default=1.0)
 4* RIN(I,ISZ) X or R fine-mesh boundaries (#=IMSISM+ISM)
 5* ENER(IG) top neutron energy group boundaries + bottom energy
 of last neutron group + top photon energy group boundaries +
 bottom energy of last group (IGM+2) [NTFOG>0]@
 6* DB2Z(IG,IZ) DB² by group, then by region (#=IGM*NREG)
 [IFDB2Z>0]
 7\$ ITHYG(IG) theory by group (#=IGM) [IGTYPE>0] @
 8\$ IJZN(I,J) material zone by fine space mesh (#=IMSJM!)@
 9\$ IZMT(IJZN) material number by material zone (#=IZM)
 10\$ MIXT(MIX) mixture ID (#=MIXL) @
 11\$ NUCL(MIX) nuclide ID (#=MIXL)
 12* DENS(MIX) number density (#=MIXL)
 13\$ MATL(MT) ID number by material (#=MTM)
 (Default MATL(MT)=MT)
 14* ZNTSR(JNTSR) Z or θ boundary positions for J-boundary source
 input (#=NJNTSR)
 15* RNTSR(INTSR) X or R boundary positions for I-boundary source
 input (#=NINTSR)
 16* ZNTFX(JNTFX) Z or θ boundary positions for J-boundary flux
 output (#=NJNTFX)
 17* RNTFX(INTFX) X or R boundary positions for I-boundary flux
 output (#=NINTFX)
 18* FJSRZ(JSZN) J-Super zone search fraction (#=NJSZN)@
 [KTYPE=4]
 19* FISRZ(ISZN) I-Super zone search fraction (#=NISZN)@
 [KTYPE=4]
 20* ABDOL(IG,J) left boundary albedo (#=IGM*JM) [IBL=5]
 21* ABDOR(IG,J) right boundary albedo (#=IGM*JM) [IBR=5]
 22* ABDOB(IG,I) bottom boundary albedo (#=IGM*IMA) [IBB=5]

23* ABDOT(IG,I) top boundary albedo ($\# = \text{IGM} * \text{IMA}$) [IBT=5]
 24* EPSBZ(IZ) flux error importance by material zone ($\# = \text{IZM}$)
 [if IEPSBZ>0]@
 25\$ ICMAT(IAC) material to be used in activity calculations
 ($\# = \text{IACI}$)@
 26\$ ICPOS(IAC) cross section table position for activity
 ($\# = \text{IACI}$)@
 27* ACMUL(IAC) activity multiplier ($\# = \text{IACI}$)@
 28\$ ITMBG(IG) initial iteration limit by group ($\# = \text{IGM}$) [IFXMI<0]@
 29\$ KEYAJ(NKEY) J positions of key fluxes in ascending order
 ($\# = \text{INKEYFXI}$)@
 30\$ KEYAI(NKEY) I positions of key fluxes in ascending order for
 like J ($\# = \text{INKEYFXI}$)@

(terminate block with "T")

F. External Boundary Source Input [INGEOM<20 and IBL,IBR,IBB, or IBT=4]

91* SII(M,J) ($\# = \text{MMA} * \text{JM}$) [IBL=4 or IBR=4]
 I boundary source for a group
 92* SJI(M,I) ($\# = \text{MMA} * \text{IMA}$) [IBB=4 or IBT=4]
 J boundary source for a group

(terminate block with "T") [T required if any arrays are required]

Left and right sources are intermingled in the same array according to direction. If $\mu > 0$, the source applies to the left boundary; if $\mu < 0$, to the right. Likewise, if $\tau > 0$, a J-boundary source applies to the bottom; if $\tau < 0$, the source applies to the top boundary. If any blocks of this type are required, IGM are required.

If IADJ>0, the energy groups must be entered in reversed order. Omit block and delimiter if INGEOM $\geq$ 20, or if no arrays are required.

G. Flux Guess Input [INPFXM>0]

93* FIJ(I,J), FIJ(I,J) or FI(I) as INPFXM=1, 2, or 3 ($\# = \text{IMSJM}, \text{IMSJM},$
 or IM) [INPFXM>0]
 94* FJ(J) ($\# = \text{JM}$) [INPFXM=3]
 95* FG(IG) ($\# = \text{IGM}$) [INPFXM>1]

Each array of this block must be followed by "T". If no arrays are required, no "T" is required. If IADJ>0, energy groups must be specified in reversed order.

H. Distributed Source Input [INPSRM>0]

96* as 93*
 97* as 94*
 98* as 95*

5.2. Additional Input Data Details

5.2.1. Special Geometry Features

If INGEOM $\geq$ 10, a one-dimensional problem will be solved with vertical flow suppressed. Use void boundary condition at top and bottom. RZ direction sets must be used with INGEOM=11, while R θ sets must be used with INGEOM=12. Solutions with INGEOM=11 or 12 yield essentially the same result.

If INGEOM=20, KTYPE=0, and IBR=4, a combined reflection/transmission problem in slab geometry is solved. Omit the 91* and 92* array blocks, and supply the input spectrum as CHI. The code will generate a boundary source in the leftmost direction of non-zero weight of each downward η level. The value of JM must be the number of downward η levels. In this case, J corresponds to the incident η level, not to a Y-direction mesh. The 2* array should be filled with the values 0.0, 1.0, 2.0, ... JM. The first η level will correspond to the last J level. |CHI(IGM)| particles will enter each level in each group. If CHI(I)<0, the emerging flux is printed by direction, then by space interval.

If INGEOM=30, the calculation proceeds as with INGEOM=20, except that emerging flux will be calculated for a source entering each group JDIRF<JJI<JDIRL in turn, and leaving in all groups IGI<IGO<IGM. The total source per group in each J-level is |CHI(IGI)|. If CHI<0, the flux from sources entering that group will be printed. The problem is "double differential," in that the flux leaving each group resulting from a source in each individual input group is calculated.

If 3<INGEOM $\leq$ 6, and if all iterations are to be performed in diffusion theory, equilateral-triangular geometry is available. Geometry options, together with the required value of IM, are:

INGEOM	TRIANGULAR GEOMETRY OPTION	VALUE OF IM
3	180 -360° Symmetry	User's option
4	60° Symmetry	2*JM-1
5 (not oper)	90° Symmetry	2*JM+1
6	120° Symmetry	2*JM

The resulting geometry is illustrated in Fig. 5.1.

The mesh input data involve certain unusual requirements. In most cases, the value of IM is a function of JM as shown above. Also, the radial dimensions used for input do not correspond directly to the actual dimensions of the mesh. For all but 90° symmetry, every mesh interval is a equilateral triangle with sides having a length S.

Fig. 5.1. Triangular Geometry Options.

JM+1 J-interval boundaries are entered, beginning with 0, and with spacing $\sqrt{3} S/2$. With 120° , 180° , or 360° symmetry, IM+1 I-interval boundaries are entered, beginning with 0, and with spacing $S/2$. With 60° symmetry, the code will automatically set IM to negative, and require the entry of JM I-sets. The j th I-set has $2j$ boundaries with spacing $S/2$. The description applicable to 90° symmetry is not available.

For all cases, the (1,1) mesh interval is a triangle with vertex down. Example meshes are shown for the various triangular geometry options.

For 60° symmetry, the parameter ISBT should be set equal to JM, and the bottom boundary condition must be reflected. For 120° symmetry, the left and bottom boundaries must be periodic if periodic boundary conditions are selected.

5.2.2. Special Theory Options

If IGTYP > 0, then the first IGTYP outer iterations will be performed using alternate theory specified by group in the 7\$ array. If the g th entry is N_g , then the first $|N_g|$ inner iterations of the first IGTYP outer iterations on group, g , will be performed using diffusion theory if $N_g > 0$. If $|N_g|$ is less than the applicable maximum number of iterations, the iterations will be completed in transport theory, even though the alternate theory may have achieved convergence.

5.2.3. Cross Section Input and Mixing

The internal cross section storage comprises MTM sets of cross sections for each energy group. The first MTP sets are read from logical unit NTSIG, which is always required. The remainder are prepared using a "mixing table," described below. [It is intended that the first MTP "materials" will be microscopic nuclide data, and the remainder will be macroscopic mixtures, although other uses are possible.] The material numbers entered in the IZMT array (9\$) must be integers between 1 and MTM, corresponding to the appropriate data.

Each material consists of IHP cross sections as indicated in the description of the ORDOSW input file. If ISCTM>0, then ISCTM sets of Legendre expansion data must follow each set designated in the IZMT array. If MCR<0, then the required number of expansion sets is indicated in the NSIG array (78\$) (this feature is not presently operable).

If MIXL>0, cross sections are to be modified by a mixing table, specified by the MIXT, NUCL, DENS, and MATL arrays (10\$, 11\$, 12*, and 13\$). The MATL array assigns an arbitrary ID number to each material. If it is not entered, the ID's 1,2,..., MTM are assumed. Each integer entered in the MIXT and NUCL arrays must be one of these ID's, or 0. In the following:

$m = \text{number of the material having ID } |p| \text{ if } p \neq 0; \text{ otherwise } 0$

$n = \text{number of the material having ID } |q| \text{ if } q \neq 0; \text{ otherwise } 0$

and the interpretation of the table is as follows:

$$\begin{array}{ccc} \frac{MIXT}{p} & \frac{NUCL}{q} & \frac{DENS}{d} \end{array}$$

- (a) If $q=0$, then the data of material m will be multiplied by d .
- (b) If $q>0$, then the data of material n , multiplied by d , will be added to that of material m .
- (c) If $q=p$, the data of material m will be modified by the eigenvalue in concentration searches.
- (d) If $p<0$, the appropriate Legendre expansion components will be treated as was the principal set.

Each set of entries is executed in sequence. In searches, the table may be executed repeatedly. As an example, with MIXL=6, MTM=11, and ISCTM=3.

I	MIXT	NUCL	DENS	MATL
1	1000	0	0	10
2	1000	10	.1	20
3	1000	20	.2	30
4	1000	1000	0	40
5	-2000	0	0	50
6	-2000	30	.3	60
7				1000
8				2000
9				3000
10				4000
11				5000

Material 7 will consist of:

Material 1 * .1 + material 2 * .2

and will be modified by the eigenvalue.

Materials 8, 9, 10, 11 will be:

Materials 3, 4, 5, and 6 * .3.

Although the value of σ^A does not affect the flux in a calculation directly, it must be used to obtain correct balance tables. It should meet the condition:

$$\sigma_g^T = \sigma_g^A + \sum_{g'} \sigma_{g \rightarrow g'}$$

If σ_g^T is replaced by a "transport cross section" in reactor core problems, $\sigma_{g \rightarrow g'}$ must be reduced such as to maintain σ^A constant.

It may be important to note that many standard cross section files such as ISOTXS and MATXS have Legendre expansion data which must be multiplied by $2l+1$, where l is the expansion index, for use in DOT 4. It is intended that this be done in the code which prepares the ORDOSW (or optional GIP input file).

5.2.4. Activity Edit

A very general provision for obtaining energy-integrated reaction rates is provided. The arrays ICMAT, ICPOS, and ACMUL define an "activity table," interpreted as follows:

$$\begin{array}{ccc}
 \frac{ICMAT}{r} & \frac{ICPOS}{s} & \frac{ACMUL}{t} \\
 A_{IJ} & = & d_{IJ} C_{IJrs} \sum_s \phi_{sJJ} \sigma_{srs} \\
 B_i & = & \sum_{IJ} X_{IJ,s} A_{IJ} V_{IJ}
 \end{array}$$

where

- i,j = space mesh indices
- g = energy group index
- lrl = material number
- lsl = cross section table position number
- t = arbitrary multiplier
- d = density factor if used, else 1
- ϕ = flux
- σ = cross section
- C = the "number density" of material lrl in space cell i,j
- X = 1 if cell i,j is in edit region Z , else 0

The value of C is determined as follows:

- (a) If r is the macro material used in cell i,j , then $C = 1$,
- (b) If r is 0, then $C = 1$, and the cross sections for the macro material in material in cell i,j are used,
- (c) If $r < 0$, then $C = 1$,
- (d) If $r > 0$ and is not the macro, then C is the sum of all assignments of r to the macro according to the mixing table. Suppose $IAC T = 4$ and the following table is entered, referring back to the mixing table illustrated above; and suppose material 7 is named as a macro in a portion of the problem:

I	$ICMAT$	$ICPOS$	$ACMUL$
1	7	3	1.
2	-7	3	1.
3	0	4	500.
4	1	5	1.
5	1	-5	1.

Activity 1 will use position 3 of material 7, but will be 0 where 7 is not the macro. Activity 2 will be similar, but will be calculated whether 7 is the macro or not. Activity 3 will use position 4 of the macro in each space cell, and will multiply the result by 500. Activity 4 will use position 5 of material 1 and a "number density" of .1 wherever 7 is the macro, since .1* material 1 was added to material 7 in the mixing table.

As a special feature, if $lsl > IHP$, then the value of CHI (I^*) for the group will be used to replace σ , and c and d will be everywhere 1. This can be useful in adjoint problems.

If $s < 0$, the activity will be written on the punch data set. This can be used for fission density output, for example, as in Activity 5 of the illustration.

5.2.5. Negative Source Removal

Any problem using $ISCTM > 0$ may incur negative source generation. This is simply due to the failure of a truncated P_L expansion to describe the cross section adequately. If

the problem has no areas of monodirectional flow or void, this effect will probably not be noticed, and will usually cause no harm. Voids or monodirectional flow, however, can result in severe distortions, or even negative scalar values for the flux. The `NEGFIX>0` option will remove these distortions in most cases. The penalty in CPU time is usually in the range of 15-30%, although it can be higher if a major portion of the problem is so affected.

5.2.6. Problem Type Specification

Five major types of calculation are available, chosen by `KTYPE`:

- `KTYPE = 0` M calculation - fixed source or subcritical multiplication
- `= 1` K calculation - determines K_{eff}
- `= 2` B search - modify DB^2 to produce specified K_{eff}
- `= 3` C search - modify concentrations to produce specified K_{eff}
- `= 4` D search - modify dimensions to produce specified K_{eff}

When `KTYPE=0` and `CHI=0` (1*array) or `NOFIS=2`, a simple fixed-source problem is solved. Flux from the input source is determined in one source iteration unless upscatter cross sections are used. Upscatter requires `ISRMX>1` and `EPO>0` for satisfactory results.

If some values of `CHI` $\neq 0$ and `NOFIS` $\neq 2$, fissions resulting from the fixed source and the input flux guess, if any, are calculated in the first source iteration. These are added to the fixed source to begin the second iteration, and so on until subsequent fission generations have been accounted for. Since these subcritical multiplication problems tend to be slowly convergent, source iteration acceleration, discussed elsewhere, is used. If the system is, by itself, supercritical, this calculation will surely fail; a reflection of the fact that the flux would be divergent in such a physical situation. The number of fission neutrons produced per source neutron is determined by iteration. Several source iterations are required. When `KTYPE=1`, a calculation of K_{eff} is performed. Acceleration of the source iterations is discussed elsewhere.

K calculations can be improved somewhat by setting `EVI` equal to the best-available estimate of `K`. This is especially important in the case of restarts.

When `KTYPE>1`, properties of the system configuration are changed in order to drive K_{eff} , calculated as described above, to a specified value, `EKOBJ`. For each configuration, the value of K_{eff} is partially converged, the configuration is changed, and the process continues until K_{eff} has stabilized and it is near the objective value. Additional details of this process are given in another section.

5.2.7. Iteration Limits and Convergence

In general, source iterations will be performed until the time `TMAX` is exceeded (`IXX=8`), the iterations limit `ISRMX` is exceeded (`IXX=1`), or convergence is reached. Source iteration convergence has not been reached if:

- (`IXX=2`) the problem uses diffusion-theory source iterations followed by transport theory, and transport theory has not begun
- (`IXX=3`) the search has not converged; i.e., `KTYPE>1`, `EPS>0`, and

$$\epsilon^s = \left| \frac{K^N - EKOBJ}{K^N} \right| > EPS$$

where: K^N is the value of K_{eff} for the latest (Nth) source iteration
 (IXX=4) flux iterations for some group have not converged;
 i.e., $EPO > 0$ and

$$\epsilon^F = \left| \frac{\phi_{i,j,g}^N - \phi_{i,j,g}^{N-1}}{\phi_{i,j,g}^N} \right| > EPO$$

(IXX=5) fission density has not converged; i.e., $EPF > 0$ and:

$$\epsilon^D = \left| \frac{F_{i,j}^N - F_{i,j}^{N-1}}{F_{i,j}^N} \right| > EPF$$

(IXX=6) K_{eff} has not converged; i.e., $EPS > 0$ and:

$$\epsilon^K = \left| \frac{K^N - K^{N-1}}{K^N} \right| > EPS$$

(IXX=7) No source iterations have been completed

The value of IXX is printed at each iteration, together with each of the above measures of convergence.

In general, problems with $KTYPE > 0$ require several outer iterations for convergence, while problems with $KTYPE = 0$ require only 1. In the latter case, performing several outer iterations can be very wasteful. Exceptional cases are fixed-source ($KTYPE = 0$) problems with non-zero fission or with upscatter. In these cases, several outer iterations are required.

The flux iteration limit for each source iteration will be IFXMI or, if $IFXMI < 0$, the groupwise value from input array 28\$. After K^N and fission density have converged, the iteration limit is changed to IFXMF if $IFXMF > 0$. Flux iterations proceed until the time TMAX is exceeded, the iteration limit is exceeded, or convergence is reached. Flux convergence has not been reached if:

- Pointwise flux convergence has not been reached; i.e., $EPP > 0$ and:

$$\epsilon^P = \left| \frac{\phi_{i,j,g}^n - \phi_{i,j,g}^{n-1}}{\phi_{i,j,g}^n} \right| W_{i,j} > EPO$$

n = flux iteration index

Where W_{ij} is the value of the importance function (24* array) for the position ij if $IEPSBZ > 0$, and otherwise 1.0. The largest ϵ^P for any group after the last flux iteration is called "group convergence," ϵ^G .

- Volumetric flux convergence has not been reached; i.e., $EPV > 0$ and:

$$\frac{\sum_{ij} \left| \phi_{ij,g}^n - \phi_{ij,g}^{n-1} \right| W_{ij} V_{ij}}{\sum_{ij} \phi_{ij,g}^n W_{ij} V_{ij}} > EPV$$

Fluxes less than FLXMIN are not considered in the tests. This prevents values near zero, which are badly affected by roundoff, from thwarting convergence. FLXMIN also is used as a minimum coarse-mesh flux by the flux acceleration routines, preventing similar degeneracy there.

5.2.8. Search Strategy

The searches proceed by successive adjustment of the eigenvalue, EV.

EV is constrained to be positive, and extensive checks prevent oscillation and overextrapolation. Each search seeks a value of EV which adjusts K_{eff} to a input value, EKOBJ. EV is used to adjust the system parameters as follows:

B search: $(DB^2)_{eff} = (DB^2)_{input} * EV$

C search: $(C)_{eff} = (C)_{input} * (1 + f(EV - 1))$

To adjust material m in this way, an entry is made in the mixing table:

$$\frac{10\$}{m} \quad \frac{11\$}{m} \quad \frac{12*}{f}$$

D search: $(\Delta d)_{eff} = (\Delta d)_{input} * (1 + f(EV - 1))$

Here, d represents either dimension and f is the superzone search fraction input (18* or 19* array)

The initial estimate of EV is EVI. Defining search convergence as:

$$CONVSR = (K^N - EKOBJ)/K^N$$

and eigenvalue convergence as:

$$CVK = (K^N - K^{N-1})/K^N$$

the initial value of EV is used until:

$$CVK < EVTH * \max(EPS, \min(0.1, CONVSR))$$

The second value of EV is determined by linear extrapolation if $EVDELK = 0$, using the input value of eigenvalue dependence on K, DEVDKI. If $EVDELK \neq 0$, the second EV is:

$$EVI \pm EVDELK$$

where the sign of DEVDKI is used to determine the correct step direction. Subsequent values of EV are obtained by nonlinear interpolation if possible, and linear when nonlinear fails.

Each new value of EV is constrained to the range between $EV \cdot EVCHM$ and $EV/EVCHM$. EV is always limited to the range between $EVI \cdot EVMAX$ and $EVI/EVMAX$. If EV moves to its limit and stays there, or if K exceeds the range between $EKOBJ + EVKMAX$ and $EKOBJ - EVKMAX$, the search is stopped.

5.2.9. Upscatter Rebalance

If upscatter is present in a problem, upscatter rebalance is performed after source iteration 2 and successive iterations. All fluxes within the upscatter energy range are multiplied by a factor DVUPS:

$$DVUPS = UPSS / (UPSS - (UPSP - UPS))$$

where: IGUPS = first group into which upscatter is non-zero

UPS = previous upscatter sum

UPSP = new upscatter sum

UPSS = sum of all scatter into groups $\geq$ IGUPS from groups $<$ IGUPS

This factor is applied to fluxes for all energy groups $\geq$ IGUPS, bringing the system into global balance. This procedure is usually helpful, and has not been observed to cause erratic behavior. The value of $\epsilon'' = DVUPS - 1$ is printed in the output data as upscatter convergence.

5.2.10. Fission Rescaling

An approach analogous to upscatter rebalance is used when a fixed source is used with a multiplying medium. In these cases, fluxes for all energy groups are multiplied by FSNORM:

$$FSNORM = FXSUM / (FXSUM - (SRNEW - FSOLD))$$

where: FXSUM = total fixed source

SRNEW = new fission source estimate

FSOLD = previous fission source estimate

Since the upscatter rebalance is applied before fission sums are calculated, there is no conflict between these procedures, and one supplements the other.

5.2.11. Fission Extrapolation

After these rebalance and rescaling procedures, a feature is provided to use a fixed fission acceleration factor:

$$(FSNIJ)_{\text{extrap}} = (FSNIJ)_{\text{calc}} + f [(FSNIJ)_{\text{calc}} - (FSOIJ)_{\text{final}}]$$

where: FSNIJ = new fission distribution

FSOIJ = previous fission distribution

f = FSNACC, an input parameter, adjusted downward as necessary to insure that no fission value is changed more than 20%

No extrapolation of fluxes is attempted with this procedure. Tests have shown that reasonably chosen values of FSNACC speed the convergence of certain problems, especially M calculations, and cause oscillation in others. Its use is not generally recommended.

5.2.12. Error-Mode Extrapolation

To accomplish error-mode extrapolation, DOT 4 accumulates the sum FERAH and the eigenvalue estimate λ :

$$FERAH = \sum (FSNIJ)_{\text{extrap}} - (FSOIJ)_{\text{extrap}}$$

$$\lambda = (FERAH)_{\text{new}} / (FERAH)_{\text{previous}}$$

and an error-mode removal parameter θ :

$$\theta = \lambda / (1 - \lambda)$$

When estimates of θ in three successive iterations are in fractional agreement within EXTRCV, the fission is extrapolated according to:

$$(FSNIJ)_{\text{final}} = (FSNIJ)_{\text{extrap}} + \theta' [(FSNIJ)_{\text{extrap}} - (FSOIJ)_{\text{final}}]$$

where θ' is θ adjusted so that the fission density will change by no more than |SORMINI|, where SORMIN is a input value. The fluxes are similarly extrapolated by the factor θ' , with each flux limited by the |SORMINI| ratio.

Tests show this procedure to be safe and effective. It is compatible with upscatter rebalance and can be used together with fission rescaling.

If fission rescaling is not used, θ , in certain cases, may be 100 or larger. In such cases, single-precision IBM calculations may not be sufficient to provide a stable value of θ , and so the extrapolation is not effective. Rescaling may lower θ in such cases to a manageable value, i.e., to 25 or less.

As a counterexample, if a system is driven by a source at its periphery, the unmultiplied flux due to the source is so different in shape from the final flux solution that fission

rescaling may slow or thwart convergence. A negative value of SORMIN must be used in such cases, so that fission rescaling is disabled.

5.2.13. Flux Iteration Acceleration

Five types of flux iteration acceleration can be selected:

IACC	KALF	Result
0	any	Groupwise rebalance
1	0	Diffusion acceleration, corner mesh
1	1	Diffusion acceleration, centered mesh
2	0	(Conventional) Partial current rebalance, J stabilized
2	1	(Conventional) Partial current rebalance, ϕ stabilized

Corner-mesh diffusion acceleration (CNRDA) is usually best for "normal" problems. High-energy neutron groups in problems having complex geometry may be better solved by partial-current rebalance (PCR). The choices of stabilization with PCR are of roughly equal merit. In the case of problems having two facing reflected or periodic boundaries, such as cell or R θ problems, it is often helpful to use the coarse-mesh feature to specify only one interval in the reflected (or periodic) direction, and to specify PCR. (In such a case, the code will arrange one interval as a default unless the 85*/86* array is entered explicitly.)

All of the flux iteration acceleration schemes except the groupwise rebalance require the performance of a certain number of rebalance iterations after each flux iteration. Default input parameters constrain this number between 4 and 100 (INTSCL and ITMSCL). In 1-D problems, or if the coarse mesh has only 1 interval in either direction, INTSCL=1 is appropriate. In K calculations and searches, a lower value of ITMSCL (e.g. 8) can lower costs. Within the iteration limits, the iterations are terminated when:

$$\text{MAX}_{ij} \left| \frac{\phi_{i,i}^K - \phi_{i,j}^{K-1}}{\phi_{i,j}^K} \right| < \text{CONSCL}$$

where K is the iteration index.

If the upper iteration limit is reached without satisfying CONSCL, the improved data are accepted for use if convergence has reached CONACC. Otherwise, unaccelerated data are used.

It is often effective to make CONSCL less than EPP by roughly a factor of 10, and to leave CONACC at its default value. Other combinations are often used, however.

If IBFSCL>0, rebalance iterations are bypassed on the first flux iteration of the first source iteration. This sometimes saves CPU time on very difficult problems, and sometimes prevents certain types of early iteration failure. On less difficult problems, especially those using diffusion acceleration, it may lower efficiency slightly.

If $WSOLOI > 0$, flux acceleration will not be performed on those energy groups for which, for all materials of a problem:

$$\frac{\sigma^S}{\sigma^T} < WSOLOI$$

This can save unneeded CPU effort without impairing convergence rate.

5.2.14. Stabilization

The partial-current rebalance method requires choice of a "stabilization factor," F . The effect of this factor is to adjust the rebalance matrix in such a way as to insure positive factors if $F > 1.0$. The user inputs an initial value, $WSOLCN$, and an increment, $WSOLII$. If $WSOLII > 0$, F is increased by $WSOLII$ after each rebalance. If $WSOLII < 0$, F is increased by $|WSOLII|$ only when incipient oscillation is detected.

The choice of these factors has a very large effect on the running time of difficult problems. Some combinations are shown in Table 5.1. If diffusion acceleration can be used, this difficult choice is avoided.

Table 5.1. Typical Stabilization Values

WSOLCN	WSOLII	Comments
1.0	0	Satisfactory for many well-behaved core problems which are probably better solved by diffusion acceleration.
1.0	1.0	Satisfactory for many difficult fixed-source problems
4.0	1.0	One solver of very difficult problem uses this
1.0	-1.5	This may be a "best-buy" single choice.

5.2.15. Input Source and Flux Specification

For K_{eff} and configuration search problems, the initial source is generated internally based on the fission cross sections and the flux guess. For fixed-source problems, the user must supply an external source of particles. This can be done by supplying an external source file containing an internal or external boundary source, a distributed source, or a first-collision source. The external boundary source and distributed source can be generated internally by card-image input in a few simple cases. At least one non-zero source type must always be supplied.

An external boundary source is indicated by the appropriate choice of boundary condition. If input parameter $NTBSI > 0$, the logical unit $NTBSI$ is to contain the information.

Otherwise, it is to be found in the card-image input. Formats for these inputs are explained elsewhere in this region.

A distributed source is to be supplied on logical unit NTDSI if NTDSI>0 and INPSRM=0. If INPSRM>0, NTDSI must point to a scratch device onto which a source generated from card images will be placed.

A first-collision source is to be supplied on NTFCI if NTFCI>0. This source is simply a special type of distributed source, and is specified by a special type of distributed source file format. In such a case, the file also gives the "uncollided flux" the scattering of which will lead to the source. This flux is added to the flux calculated by the code in some of the output results. (It is subtracted from a flux guess on NTFLX, if any.)

Internal boundary sources are indicated by NJNTR>0 and/or NINTSR>0. In these cases, internal boundaries are specified in the 14* and 15* arrays at which sources input from logical unit NTIBI are located.

In principle, several types of sources can coexist. Little actual experience is available in these areas, however, and such results should be carefully checked.

If KTYPE>0, a non-zero flux guess is always required. If NTFLX>0 and INPFX=0, the flux is to be supplied on logical unit NTFLX. If INPFXM>0, the flux guess is to be generated from card images. Problems with KTYPE=0 are usually started without a flux guess, unless the problem has been partially solved in a previous execution.

5.2.16. Key Flux Monitoring

In many cases, the convergence of all of the flux data is of less interest than convergence of one or a few key fluxes. If KEYJN>0 and KEYIN>0, the value of a single key flux will be printed in the flux-iteration monitor line. If not, the position of maximum flux change at the second flux iteration will be so monitored. In addition, an arbitrary array of key fluxes can be specified by adjusting NKEYFX. If desired, these fluxes can be printed only after the convergence of each energy group.

5.2.17. Energy Boundary Input

The energy boundaries are used only for transfer to the output files. They may be ignored for most applications. The ordering is:

Combined Neutron-Photon	Neutron Only	Photon Only
E_1^{MAX}	E_1^{MAX}	0
E_2^{MAX}	$\vdots$	E_1^{MAX}
$\vdots$	$\vdots$	$\vdots$
$\vdots$	E_{IGM}^{MAX}	$\vdots$
E_{NEUT}^{MAX} (last neutron group)	E_{IGM}^{MIN}	E_{IGM}^{MAX}
E_{NEUT}^{MIN}	0	E_{IGM}^{MIN}
E_{n+1}^{MAX} (first photon group)		
$\vdots$		
E_{IGM}^{MAX}		
E_{IGM}^{MIN} (last photon group)		

5.2.18. Flux Extrapolation Model

The choice of mode can have a important influence on both CPU time requirements and the accuracy of the solutions. In general, the θ -weighted model gives reasonable results for a wide range of problems. The linear-zero model is similar to the tried-and-proven method used in TWOTRAN-II. It gives well-accepted and rapid solutions to problems such as a reactor core not containing severe heterogenities. It will often fail to converge or give inaccurate results for difficult deep penetration problems, however. The ordinary weighted model almost always converges better than the other models. Its results are usually not suitable for K_{eff} calculations, but it is used exclusively by many deep-penetration analysts.

5.2.19. Alternate Zone Map Input

An alternate "combinatorial-geometry" option is available. Rather than the usual 85 description, supply the following:

$-z_1 \quad i_1 \quad j_1 \quad I_1 \quad J_1$
 $-z_2 \quad i_2 \quad j_2 \quad I_2 \quad J_2$
 etc
 F 0

Each set of five integers contains:

-z: The zone number, tagged with a negative
 $\left. \begin{array}{l} i \\ j \end{array} \right\}$ Indices of lower, lefthand corner of zone in zone map
 $\left. \begin{array}{l} I \\ J \end{array} \right\}$ Indices of upper, righthand corner of zone in zone map

As many descriptors as required to a limit of IMSJM are used, and the remainder of the array is filled with 0. Zones can be overlaid as desired. This type of 8\$ array must not be intermingled with the traditional type within a given problem. When used with variable mesh, the zones are understood to be cylindrical in space, although they may be discontinuous in the zone map printout.

5.3. Details of Job Requirements

5.3.1. CPU Time Usage

A reasonably accurate time limit should be calculated and entered as TMAX for all problems which run more than a few minutes. The maximum time given to the operator or operating system should be somewhat larger than TMAX. This allows DOT 4 to complete its I/O and data summaries if the time is exceeded during the iteration phase. The CPU time required, C, can be estimated by the following formula:

$$C = A + \left\lceil \frac{FW}{FR} \right\rceil * \frac{150}{SF}$$

$$FW = \sum_P \sum_G \sum_F \sum_J \sum_I MM$$

where

FW = flux work
 MM = number of directions
 I,J = space-mesh indices
 F = inner (flux) iteration index
 G = energy group
 P = outer (source) iteration index
 FR = flux rate on the reference machine, the IBM 370/3033
 SF = the speed factor for the machine used
 A = overhead (I/O, problem setup, and summary activities)

The value of A is typically 0 to 2 minutes on the IBM 360/91. Suggested speed factors are shown in Table 5.2.

Table 5.2. Speed Factor
(Based on IBM 709 = 1)

CDC 7600	250
IBM 360/195	250
IBM 370/3033	150
IBM 360/91	150
CDC 6600	30
IBM 7094	10
IBM 7090	5

The value of FR depends somewhat upon the class of problem solved. FR is about 1.0 million fluxes per minute for large P_3 problems using the weighted difference model. The linear-zero model sometimes runs a bit faster, and a P_0 calculation runs as much as 40% faster. The IBM speeds assume the use of optional assembler language flux routines. Assembler language routines are not available for CDC use. Expert opinions offered to us differ as to how much the CDC performance would benefit from the use of assembler language.

5.3.2. Input and Output Data Files

All of the user input and output data files are expressed in formats designed according to recommendations of DOE's Committee on Computer Code Coordination (CCCC).³⁹ The specifications of these files are contained in Appendix B. Two exceptions, retained for backward compatibility, are an optional capability to process a "GIP" cross section input file and to produce a "DOT Angular Flux Tape" as output.

The GIP file is defined exactly as the records labeled "CROSS SECTION DATA" in the ORDOSW file, and has no other record types. The "DOT III Angular Flux Tape" is defined in the DOT III User's Manual,¹¹ and is simulated by DOT 4 as accurately as possible. An alternate format is used when the albedo generation option is used.⁴¹

The correspondence of file type to use in the code is:

SYMBOL	FILE TYPE	USE
NTFLX	VARFLM	Flux and moment input
NTFOG	VARFLM	Flux and moment output
NTSIG	ORDOSW or GIP	Cross section input
NTBSI	BNDRYS	External boundary source input
NTDSI	VARSOR	Distributed source moment input
NTFCI	VARSOR	First collision source input
NTIBI	BNDRYS	Internal boundary source input
NTIBO	BNDRYS	Internal boundary flux output
NTDIR	DOT III "Angular Flux Tape"	Directional flux output
NTDSO	VARSOR	Directional flux moment output

5.3.3. Scratch Data Sets

The following lists the scratch data sets, the length of each record, and the number of records:

Logical Unit Number	Logical Unit Symbol	Description	Record Length l rec	No. of Records nrec	Required
81		Scalar Flux Scratch	IMA*JM	IGM	Not Used at this time
82	NDSIG	Cross-Section Scratch	IHM*MTM	IGM	Always
83	NDBTL	Balance Table Scratch	NREG*10+IZM+8	IGM	IREDA ≠ 0
^a 84	NDBSI	Boundary Source Scratch	(IMA+JM)*MMA (IMA*NJNTSR+JM*NINTSR)*MMA +ICMJCM+NREG	IGM	NTBSI>0 (NINTSR>0 or NJNTSR>0)
91	NDFLX	Flux Scratch	IMA*LM*NJBLK1	NJBLK*IGM	NJBLK>0
92	NDSOR	Total Source Scratch	IMA*LM*NJBLK1	NJBLK	NJBLK>0
93	NDSIN	IN-Source Scratch	IMA*LM*NJBLK1	NJBLK	NJBLK>0
94	NDBFX	Boundary Flux Scratch	(IMA+JM)*MMA	IGM	Always
95	NDFIJ	Scalar Flux Scratch	IMA*JM	IGM	Always

NOTE: The symbol "J" denotes the word "number."

JBLK1 = #J levels per J block, $1 \leq JBLK1 \leq JM$

NJBLK = #J blocks, $MINBLK \leq NJBLK \leq MAXBLK$ (NJBLK=0 if fluxes are stored in working memory)

LM = maximum number of moments (including $O(h)$) in spherical harmonic expansion, $LM = 1 + \frac{1}{2} ISCT \cdot ((\frac{1}{2} ISCT + 3)) / 2$

NREG = number of edit regions

IZM = number of material zones

^aIf NTBSI>0 and NINTSR>0, for example, this file will have IGM + IGM records.

The number of J blocks, NJBLK, is controlled by input data. When default values of MINBLK and MAXBLK are used, the code will attempt to hold all flux and source data in fast memory, i.e., NJBLK=0. Failing in this, it will attempt to allocate space for the entire mesh to be held as 1 block in the user-buffer area, i.e., NJBLK=1. Failing in this, the space mesh is broken into as many blocks as required. NJBLK is restricted to limits specified by MINBLK and MAXBLK.

5.3.4. System Buffer Space (NBUF)

In the normal IBM configuration, the space for system buffers and system control blocks used by all data sets except the card-image input and printed output must be allocated by specifying NBUF in the input parameter data. For each user-supplied data file, the space required in K-byte units (1 K-byte = 1024 bytes) is:

$$K\text{-bytes required} = 2 + 2 * \left\lceil \frac{BLKSIZE * BUFNO}{2048} \right\rceil$$

where $[]$ indicates truncation, and BLKSIZE and BUFNO are as specified in the DCB field of the JCL card. If BUFNO is not specified, assume 2.

For scratch files required as logical units 81-95, assuming $ntrk$ is the number of disk tracks required for a logical record, the space required for system control blocks for each data file is

$$K\text{-bytes required} = 4 + 2 * \left\lceil \frac{205 * ntrk}{2048} \right\rceil \quad (\text{approx})$$

On UNIVAC systems, a small amount of storage is required to hold record pointer data. For each of the scratch data files required as logical units 81-95 the requirement is:

$$K\text{-words required} = 1 + \left\lceil \frac{26 + rectntrk}{1024} \right\rceil$$

NBUF must be as large as the sum of the requirements for all data files.

Default values are satisfactory for many problems on either IBM or UNIVAC systems. NBUF is not required for CDC operation.

5.3.5. Memory Requirement -- Standard Configurations

The user must understand the various storage arrangements in order to provide enough space for data storage. An exact formula for the amount of storage required is not practical. Numerous small arrays are assigned, and arrays overlap in storage in many cases. Approximate methods have proven satisfactory, however.

Figure 5.2 illustrates the storage arrangement on three computer configurations to which DOT 4 has been adapted. On IBM systems, the data can occupy all space allocated in the JCL except that used by the actual program instructions. A region consisting of NBUF K-bytes is reserved at the top of this area for system blocks allocated by the I/O routines. Data arrays which communicate with the three flux- and source-moment units, NDFLX, NDSOR, and NDSIN are assigned to the user-buffer area, L3. All other arrays are assigned to the "fast core" array area, L1.

When NJBLK=0, L3 is not used, and the 3 scratch files which communicate with it are not opened. This will often make L1 larger than the available space. If L1 is too large, the code changes NJBLK to 1, moves the moment data to the scratch files, and reserves space in L3 for all of the moments belonging to one energy group. This step greatly reduces L1, while increasing L3 by a lesser amount.

If the storage regions are still too large, the data for a single group, stored in LBUF, are broken into NJBLK blocks, each with a maximum of JBLK J-levels. L1 is increased slightly by this action, but L3 is reduced. In the limit, NJBLK can be as large as JM, resulting in a very small L3. As a practical matter, NJBLK=1 usually causes very little impairment of efficiency. NJBLK $\geq$ 2 may cause a significant increase in I/O time on some systems.

Fig. 5.2. Problem Data Storage.

If $\text{MINBLK} > 0$, then NJBLK is required to be at least as large as MINBLK . This is used largely for testing. If $\text{MAXBLK} > 0$, NJBLK can be no larger than MAXBLK . This feature is used largely to prevent undesired levels of I/O delay. On virtual IBM systems, using a region size large enough to achieve $\text{NBLK1} = 1$ appears to be appropriate.

Table 5.3 shows the larger arrays required in the region L1, except the flux and moment storage arrays. These latter arrays are shown in Table 5.4, together with the L3 requirements. Experience has indicated that these major arrays comprise at least 75% of the total storage requirement. Thus, $4/3 \times$ these estimates is a useful guess. A short trial run will quickly clarify whether the guess was satisfactory.

The UNIVAC data array arrangement is nearly identical to the IBM arrangement. The code attempts to fit the data into a region of length LOC OBJ . If this is not adequate, limit will be extended as required, up to a limit built into subroutine ALOCAT . The user region is then extended to the required amount by ALOCAT .

Present standard CDC configurations use an "extended memory" concept. Certain large, but little used, arrays as indicated in Table 5.3 are moved to the lower portion of slow memory. This area, L2, is addressed directly by LEVEL 2 FORTRAN operations. The area L3 is addressed by block-move routines. Although this concept is not provided for in Reference 40, it has been shown to cause no measurable loss of efficiency. It is essential to practical solution of large problems on the CDC 7600. Additional arrays indicated by Table 5.5 are also assigned to L2.

The region L2 must fit within 131000 words. The code attempts to fit L1 within the limit LOC OBJ and L3 within the limit LCMOBJ . Failing this, limits for either region are increased to the maximum amount built into subroutine ALOCAT . As in UNIVAC operation, the user region is then extended to the required amount by ALOCAT .

5.4. Problem Printed Output

Each DOT 4 job starts with a notice of the date and time, followed by a record of the particular version of the code being used, and messages from the programmer informing the user of warnings or special features. The title supplied by the user as input data is next. As each array of input data is read, a notice of the array number and length is made.

The first array edits are of the 61\$ (logical unit assignments), 62\$ (integer control parameters), and 63* (real control parameters). The initial allocation of fast memory and (CDC only) slow memory are listed, followed by primary dimension-setting arrays, mixing table information, material assignment, importance, fission spectrum, and search fractions.

The next section is a set of data describing the directional remesh process, $\text{MM} < 0$. These tables are largely for the benefit of programmers in diagnosing troublesome problems. The quadrature set assignment by super-zone is an input array. The transition table indicates which transition set governs the transition from the row M-Set to the column M-Set.

The unpacking arrays follow; one for each M-Set. These integers simply indicate places in a full IMMI-direction array where data corresponding to a smaller M-Set can be kept without being overwritten.

Table 5.3. Fast Memory Arrays Needed for DOT 4

Array	Length	Needed?	Use	Moved To Slow Memory on CDC
SIG	IHP*MTM	Always	Cross sections	
ISZN	IMSJM	Always	Zone map	
FLIJ	IMSJM	Always	Scalar flux and other scratch	
FSNIJ	IMSJM	(NOFIS#2 and CHI#0)	New fission	Yes
FSOIJ	IMSJM		Old fission	Yes
FIO	MMSJM	Always	I - Boundary flux	
FJO	MMSIMS	Always	J - Boundary flux	
SII	MMSJM	L/R Bndy source	I - Boundary source	Yes
SJI	MMSIMS	T/B Bndy source	J - Boundary source	Yes
<div style="display: inline-block; vertical-align: middle;"> CPCDU ↓ R ↓ L ↓ D ↓ X ↓ SBAN </div>	ICPJCP	Always	Rebalance data (can be made small by choice of coarse mesh)	Yes
CPCF		EPP < 0	Convergence test	Yes
FIP	MMSIMS	MM < 0	Remesh scratch	
FJI	MMSIMS	(MA < 0 or IM < 0)	Remesh scratch	
FMD	<div style="display: inline-block; vertical-align: middle;"> { LVL MX*IMA MMA*(3+IMS) MMSIMS } </div>	Always NEGFIX > 0 IM < 0	Scratch for ROW Scratch for SORX Remesh scratch	
DIRF	IMA*MMDNUP	IDIRF > 0 or NEGFIX = 1	Directional flux or source storage	
FLXJT	MMSIMS*NJNTFX	Always	Internal boundary flux storage	
FLXIT	MMSJM*NINTFX	Always	Internal boundary flux storage	
SORJT	MMSIMS*NJNTSR	Always	Internal boundary source storage	
SORIT	MMSJM*NINTSR	Always	Internal boundary source storage	

- NOTES: 1. LVL MX is the largest number of n-levels in either upward or downward directional quadrature hemisphere.
 2. Multiple lengths are given for some arrays. Use the longest.
 3. MMDNUP is the largest number of directions in either downward or upward hemisphere in any direction set.

Table 5.4. Flux and Source Moment Storage

NJBLK	L1 Requirement	L3 Requirement
0	$3 \cdot \text{IMSJM} \cdot \text{LM} \cdot \text{IGM}$	0
1	$3 \cdot \text{IMA} \cdot \text{LM}$	$3 \cdot \text{IMSJM} \cdot \text{LM}$
2	$3 \cdot \text{IMA} \cdot \text{LM}$	$3 \cdot \text{IMA} \cdot \text{JBLK1} \cdot \text{LM}$

Table 5.5. Additional Extended-Memory Arrays

Array	Length	Needed?	Use
LCFXIJ	MAX MMSJM + MMSIM	Always IMSJM	Buffer to output files

The actual transition sets are given next. Each set consists of two columns of integers. Each position in each column corresponds to a direction in the corresponding M-Set. The remeshing routine decodes the integers in order to determine how to construct fluxes in a new M-Set which correspond to a previous M-Set. Since the tables are given in unpacked form, as needed by the code, unused positions are filled with "-1."

The direction sets are listed, together with "BETA" (the μ -flow coupling term), μ -mates, τ -mates, and level-mates. The μ -mate of direction (μ, τ) is direction $(-\mu, \tau)$. The τ -mate is direction $(\mu, -\tau)$. The level-mate is the first direction in the η -level containing the τ -mate. Unsymmetric direction sets may not possess all of these mates. If a mate required for a boundary condition can not be found, an error flag is set.

The values of the coupling set $C_{m,j}$ are edited, followed by the J fine-mesh and I fine-mesh sets. The real values of super-mesh and coarse-mesh boundaries are interpreted into integer assignment tables edited at this point. The total size of these meshes, number of super groups, and of edit zones are given, together with related input array edits.

Maps of the zone and material assignments are given, together with the integral values of the unnormalized input sources by group, with the total as group IGM+1. A table of storage usage gives data useful to the programmer, as well as items needed in adjusting the input data, as discussed elsewhere.

A source iteration monitor line ends each source iteration, and initial values are indicated in an "iteration 0" line. Entries on this line are:

ITN	Source iteration number
FLX	Number of flux iterations completed for this source iteration
CUM	Cumulative number of source iterations
CNV	$10 \cdot \text{IXX} + \text{IFCONV}$
	IXX as noted earlier in Section 5.2.7

	IFCONV = 0	Source iterations are to continue
	= 1	One more iteration is required
	= 2	Iterations are to stop at once
	= 3,4,5	As in 0,1,2, but configuration recalculation is required
	=	
ACC	100*ISR + 10*NEWCFG + IACX	
	ISR = 0	No eigenvalue extrapolation was performed
	= 1	Linear eigenvalue extrapolation was performed
	= 2	Nonlinear eigenvalue extrapolation was performed (2-point)
	= 3	Nonlinear eigenvalue extrapolation was performed (3-point)
	NEWCFG = 0	Old configuration is to be used
	= 3	New configuration is to be used
	IACX = 0	Extrapolation of fluxes was bypassed
	= 1	Extrapolation of fluxes was performed
IFX	I-position of maximum flux change between source iterations	
JFX	J-position of maximum flux change between source iterations	
GFX	Group of maximum flux change between source iterations	
EV	Eigenvalue edit	
K	K-effective edit	
K-CNV	K convergence, ϵ^K	
FSN-CNV	Fission convergence, ϵ^D	
FLX-CNV	Flux convergence, ϵ^F	
GRP-CNV	Group convergence, ϵ^G	
SRH-CNV	Search convergence, ϵ^S	
UPS-CNV	Upscatter convergence, ϵ^U	
EXTRAP	Fission extrapolation factor	
TIME	Time iteration was completed	

The interpretations of EV and K are:

KTYPE	EV edit	K edit
0	$\sum V \nu \sigma^F \phi$ ($\sum \chi V \phi$ for adjoint)	$\sum \chi V \nu \sigma^F \phi$
1	"	"
2,3,4	search eigenvalue	"

If the χ 's do not sum to 1, K will differ from EV in a K-calculation by a ratio equal to the sum of χ . Even if χ is precisely normalized to 1 by selecting the input value $\text{NOFIS} = 0$, EV and K may be slightly different due to roundoff.

As each flux iteration is performed on each energy group, another monitor line is printed. It contains:

GRP	Energy group number
ITN	Flux iteration number
IMFD	I-position of maximum flux change
JMFD	J-position of maximum flux change
MX FX DV	Relative maximum flux change
MX DV FX	Most rapidly changing flux value
REBL	Number of rebalance iterations - a negative value means the rebalance factors were not used
REBL ERR	Maximum relative change in rebalance factors in the last iteration
MAX REBL	Largest amount by which any factor deviates from 1
GRP REBL	Amount by which groupwise rebalance factor (whether used or not) deviates from 1
VOLM ERR	Volumetric convergence, discussed earlier, except that absolute value is not used
KEY FLUX	Value of single key flux requested by input data, after acceleration
NEGFIX	Maximum fraction of negative total source at any position, if $\text{NEGFIX} = 1$
SOURCE	Total of all sources into a group, including external boundary sources
TIME	The time at which group iteration is begun is given at the end of the heading column. Arbitrary diagnostic information is given below the time indication.

Iterations performed using diffusion theory will provide only the first six items. Edits of key flux values and/or convergence by region may follow each iteration or the set of iterations for each group, according to the options chosen. The interpretation of flux convergence also depends on options chosen, as discussed earlier.

If a dimension search has been performed, the final dimensions are edited. Flux and fission source edits follow, together with activity edits by interval and region.

System balance tables follow by group and region. Group and system totals are given as $\text{IGM} + 1$ and $\text{NREG} + 1$. The columns are largely self-explanatory. One may note that an external boundary source is external to the system, and is, therefore, included in these tables as a flow at the corresponding boundary. Flow and leakage signs are such that a positive value is outward from the region. Fission source is source used to calculate the flux, while fission rate is the fission which results from the fluxes. They will tend to differ by $K \sum \chi_g$ in a K-calculation. Inscatter and outscatter are scatter to and from the

energy group. They should, in general, be equal in neutron-only problems. This will not be true in coupled n- γ problems. The outscatter is calculated from:

$$\text{outscatter} = \text{collisions} - \text{selfscatter} - \text{absorption}$$

and will be valid only if the absorption cross section meets the balance condition.

The edit of selfscatter per collision is valuable in judging the difficulty of converging fluxes for a given energy group. In general, values larger than 0.85 may give the PCR flux acceleration difficulty. The average density factor is collision weighted.

The value of OUT/IN approaches unity for well-converged problems. It is:

$$\frac{\text{Collision} + DB^2 \text{ Loss} + \text{Positive Boundary Outflow}}{\text{Fixed Source} + \text{Fission Source} + \text{Inscatter} + \text{Negative Boundary Outflow}}$$

Sources, flows, leakage, and OUT/IN may not be correct in the case of groups in which no iterations are performed.

The fluxes, as well as balance table information which depend on them, are not divided by K_{eff} , and will, in general, tend to vary according to K_{eff} . Some users speak of this condition as "fluxes normalized according to K."

Each problem ends with a summary of clock and CPU time binned according to major sections of the code. Depending on configuration, some bins may not have labels. These are for diagnostic information only.

Some output may not appear in certain problems, depending on the options chosen. The use of NTPRT can alter the order of printing on some systems, or can transfer some output items to alternate output devices.

Region 6. Operation at ORNL

6.1. Program Access

At ORNL, the most up-to-date version of DOT 4 which is deemed satisfactory for production use is kept in an on-line data set pointed to by the catalogued procedure DOS. The member name applicable at the time of this writing is DOT4P3. Thus, to access this program, the user enters, for example:

```
//DOS EXEC DOS,REG=640K
      (JCL Describing User Data Sets)
=DOT4P3
      (User Input Data)
=END
//
```

The "EXEC" card invokes a catalogued procedure which is listed below:

```
//DOS PROC PROC=DRIVER,DSNAME=X.VARI=636.PROG=,UNIT=,VLM=,
//  PAK=,FT99=SYS1,SY=SYSDA,CYL=CYL,REG=384K,BLK=6144,GOTIME=1440,
//  PABLK=300,SABLK=300,IACYL=5,LGCYL=10,MXCYL=20,D6=A,Q90=A
//-----PREPARED BY V.A.RHOADES, 4-6104-----
//GO EXEC PGM=,PROG,TIME=,GOTIME,PAK=,PAR,REGION=,REG
//STEP18 DD DSN=,DSNAM,DISP=,SHR,UNIT=,LUNT,VOL=,SER=,VLM
//SYSPRINT U, SYSOUT=,Q90
//FT06F001 DD SYSOUT=,Q90
//FT07F001 DD SYSOUT=,B
//FT09F001 DD SYSOUT=,Q90
//FT01F001 DD UNIT=,SYS,DISP=(NEW,DELETE),
//  SPACE=(,BLK,(,PABLK,(,SABLK)),
//  DCB=(,RECL=,X,BLKSIZE=,BLK,RECFM=VBS,BUFNO=)
//FT02F001 DD UNIT=,SYS,DISP=(NEW,DELETE),
//  SPACE=(,BLK,(,PABLK,(,SABLK)),
//  DCB=(,RECL=,X,BLKSIZE=,BLK,RECFM=VBS,BUFNO=)
//FT03F001 DD UNIT=,SYS,DISP=(NEW,DELETE),
//  SPACE=(,BLK,(,PABLK,(,SABLK)),
//  DCB=(,RECL=,X,BLKSIZE=,BLK,RECFM=VBS,BUFNO=)
//FT04F001 DD UNIT=,SYS,DISP=(NEW,DELETE),
//  SPACE=(,BLK,(,PABLK,(,SABLK)),
//  DCB=(,RECL=,X,BLKSIZE=,BLK,RECFM=VBS,BUFNO=)
//FT05F001 DD UNIT=,SYS,SPACE=(,80,(,50,(,25)),
//  DCB=(,RECFM=FB,RECL=,80,BLKSIZE=,3200,BUFNO=)
//FT06F001 DD UNIT=,SYS,DISP=(NEW,PASS),DSN=,SIGNAS,
//  SPACE=(,BLK,(,PABLK,(,SABLK)),
//  DCB=(,RECL=,X,BLKSIZE=,BLK,RECFM=VBS,BUFNO=)
//FT07F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT08F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT09F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT10F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT11F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT12F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT13F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT14F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT15F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT16F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT17F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT18F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT19F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT20F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT21F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT22F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT23F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT24F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT25F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT26F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT27F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT28F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT29F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT30F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT31F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT32F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT33F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT34F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT35F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT36F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT37F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT38F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT39F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT40F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT41F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT42F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT43F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT44F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT45F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT46F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT47F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT48F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT49F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT50F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT51F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT52F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT53F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT54F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT55F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT56F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT57F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT58F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT59F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT60F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT61F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT62F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT63F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT64F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT65F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT66F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT67F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT68F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT69F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT70F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT71F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT72F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT73F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT74F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT75F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT76F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT77F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT78F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT79F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT80F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT81F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT82F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT83F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT84F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT85F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT86F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT87F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT88F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT89F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT90F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT91F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT92F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT93F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT94F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT95F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT96F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT97F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT98F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//FT99F001 DD UNIT=,SYS,DISP=(,DELETE),SPACE=(,CYL,(,DACYL,(,CONTIG))
//DOT EXEC DOS,REG=384K,GOTIME=(,30), EXECUTE JOB ON DOS DRIVER
//PENDING
```

In the illustration, the (optional) "REG" parameter was used to specify a region size (the default value is 384K). Other parameters likely to be useful include:

GOTIME	minutes of execution; preset at 1440
DACYL	number of cylinders allocated for 81,82,83,92,93,95
LGCYL	number of cylinders allocated for 84,94
MXCYL	number of cylinders allocated for 91
PABLK	primary block allocation for 1,2,3,4,8
SABLK	secondary block allocation for 1,2,3,4,8
Q6	output type for FT06F001; preset at "A"
Q90	output type for FT90F001; preset at "A"
DSNAM	specification of data set name, unit, and volume for a program data set other than the standard
UNT	
VLM	

JCL for logical units 1,2,3,4, and 8 are provided as scratch data sets for convenience in using certain programs. FT08 is given a temporary data set name and left in pass status for convenience in running certain multiple-step jobs. Several of the programs stored as members in the data set referenced by the procedure can be executed together; for example:

```
//DOS EXEC DOS,LGCYL=20
=GIP
(Input Stream to Load Cross Sections onto 8)
=DOT4P3
(Input Data for First DOT Problem)
=BNDRYS
(Input Data to Process Output From First DOT Problem As Input to Second)
=DOT4P3
(Input Data for Second DOT Problem)
=END
//
```

Load modules from other data sets can be executed together with DOT by, for example:

```
//JUNK EXEC DOS,DSNAM='I.AM',UNT=VERY,VOL=LOSTOO
//STEPLIB DD
// DD DISP=SHR,DSN=X.WAR14636.PROG
```

In this case, the standard data set has been concatenated to the user data set in order to add the driver and DOS programs to the user's pool of available programs. These specifications must precede the user's input/output data set specifications.

By setting NTNPR in the input stream, large-scale print arrays will be directed to FT90F001, rather than FT06F001. These arrays will then print at the end of the job. By setting Q90=Q, these arrays will be transferred to microfiche. By setting Q6 to Q also, all output will be transferred to microfiche.

6.2. Disk Space Allocation

The space for units 1,3,4, and 8 are calculated using rules for normal FORTRAN data sets. The JCL can be seen from the listing of the DOS procedure. The rules for the direct-access data sets 81 through 95 are quite different, however. Each logical record starts on a new track.⁴² Thus, for FT95:

$$ntrk = \# \text{ of tracks per record} = 1 + \left\lfloor \frac{4*IMA*JM-1}{N} \right\rfloor$$

where N is the byte capacity of a track, 13030 for type 3330 disks and 19069 for type 3350, and $\lfloor \rfloor$ indicates truncation to the next lower integer. The number of cylinders required for this data set is then the number necessary to contain IGM records, each record using $ntrk$ tracks. Space for these data sets should always be allocated by cylinder.

On some very large problems, the two parameters DACYL and LGCYL may not give enough flexibility in allocating space. An individual space allocation can be increased by JCL such as the following:

```
//FT94F001 DD SPACE=(CYL,200,,CONTIG)
```

This type of JCL must follow the STEPLIB specification (if used) and precede specification for user-supplied data sets.

The user is cautioned that the space required by 91, 92, or 93 does not always increase monotonically with increasing NJBLK, and a safety margin is a good precaution. Since the cylinder requirement is printed in the problem output, a short checkout run will eliminate doubt.

6.3. I/O Requests and Total Clock Time

At ORNL, the I/O request count (EXCP's) and total clock time must be estimated. The process is unfortunately complicated and imprecise. The following estimates apply:

$$R = R1 + (R2 + R3)*ntrk$$

$$T = W + C + T1 + T2 + T3$$

where:

- R = total number of I/O requests
- R1 = requests with NJBLK=0; i.e., with flux and source moments stored internally
- R2 = requests required to compute the inscatter source with NJBLK>0; i.e., with moments stored externally

R3 = requests required to compute the fluxes with NJBLK>0
 T = total clock time
 C = CPU usage
 W = wait time
 T1 = I/O time with NJBLK=0
 T2 = I/O time required to compute the inscatter source with NJBLK>0
 T3 = I/O time required to compute the fluxes with NJBLK>0
 ntrk = number of tracks per block for NDFLX, discussed previously

The determination of C was discussed earlier. The value of W depends largely on the number and type of jobs running concurrently with DOT. This cannot be predicted accurately or controlled by the user. Values of W from 10% to 80% of T are not uncommon. For jobs with NJBLK=0, R2=R3=T2=T3=0. The values of R1 and T1 are often so small for such a job that a rough estimate based on experience will suffice. Exceptions can occur, however. For example, a problem with JM=50 and IGM='00 will, if it uses a distributed source input, write 5000 small logical records on NTDSI. If the system fails to block several of these blocks into a single I/O request, a single outer iteration would log 15000 I/O requests on NTDSI for the first outer iteration, and more with additional outers. Other such exceptional cases probably exist, but a fully detailed accounting would be impractical.

When NJBLK=1, we can estimate:

$$\begin{aligned}
 R2 &= (1/2) * IGM * IGM * SI \\
 R3 &= 3 * IGM * SI \\
 T2 &= (TM * ntrk + TS) * R2 \\
 T3 &= (TM * ntrk + TS) * R3
 \end{aligned}$$

Where:

SI = number of source (outer) iterations
 FI = number of flux (inner) iterations
 TM = time required to write successive tracks
 TS = effective startup time

On the 3330 disks, TM and TS are approximately 34 ms and -17 ms. The negative starting time is due to the fact that transfer is initiated immediately when the track origin passed the disk head, but an entire rotation is lost between the writing of successive tracks. This "sector search" effect was not obtained on the 2314 disks. We do not have data for the newer 3350 units. The formula assumes that cylinder search time is relatively insignificant, which is often the case.

If another vigorous job is sharing the diskpack holding NDFLX, for example, cylinder search could add an average of 30 ms per search. The FBSAM disk package (7) chains the track-writing commands together in an attempt to prevent interruption between individual tracks of a logical record. When successful, then, the hazard of random cylinder search would occur only at the beginning of each logical record and at the beginning of each new cylinder, and thus would be comparatively small for jobs with ntrk>>1.

When NJBLK > 1:

$$\begin{aligned} R2 &= (1/2) * IGM * IGM * SI * NJBLK \\ R3 &= (3 + 11 * FI * SI) * IGM * NJBLK \end{aligned}$$

and T2 and T3 are determined as before. These procedures have given estimates of 70% to 90% of the actual I/O count in certain test problems, but may be less accurate in others. We do not know of an exact procedure which would be of manageable complexity.

The user is cautioned that computer operations expects a job to reside in memory no longer than:

$$C + \frac{RM}{1000}$$

where RM = the I/O request limit specified. Jobs which reside longer may be cancelled, and money and jobs can be lost. A sufficient value of RM appears to be the only defense against this problem.

REFERENCES

1. W. A. Rhoades, D. B. Simpson, R. L. Childs, and W. W. Engle, Jr., "The DOT-IV Two-Dimensional Discrete Ordinates Transport Code with Space-Dependent Mesh and Quadrature," ORNL/TM-6529 (January 1979).
2. K. D. Lathrop and F. W. Brinkley, "TWOTRAN-II, An Interfaced, Exportable Version of the TWOTRAN Code for Two-Dimensional Transport," LA-4848-MS (July 1973).
3. F. R. Mynatt, et al., "Development of Two-Dimensional Discrete Ordinates Transport theory for Radiation Shielding," CTC-INF-952 (August 11, 1969).
4. B. G. Carlson and K. D. Lathrop, "Transport-Theory - The Method of Discrete Ordinates," LA-3251-MS (Rev. October 22, 1965).
5. B. G. Carlson, "Solution of the Transport Equation by S_n Approximations," LA-1599 (October 1953).
6. B. G. Carlson and G. I. Bell, "Solution of the Transport Equation by the S_n Method," *Proc. U.N. Intern. Conf. Peaceful Uses At. Energy*, 2nd, Geneva P/2386 (1958).
7. B. Carlson, C. Lee, and J. Worlton, "The DSN and TDC Neutron Transport Codes," LAMS-2346 (February 1960).
8. B. Davison, *Neutron Transport Theory*, Oxford University Press, 174 (1957).
9. S. Chandrasekhar, *Radiative Transfer*, Oxford University Press, 1950.
10. G. C. Wick, "Über Ebene Diffusionsprobleme," *Z. Phys.* 121, 702 (1943).
11. W. A. Rhoades and F. R. Mynatt, "The DOT-III Two-Dimensional Discrete Ordinates Transport Code," ORNL/TM-4280 (September 1973).
12. K. D. Lathrop, "TWOTRAN - A FORTRAN Program for Two-Dimensional Transport," GA-8747 (July 1968).
13. E. T. Tomlinson, W. A. Rhoades, and W. W. Engle, Jr., "Flux Extrapolation Models Used in the DOT IV Discrete Ordinates Neutron Transport Code," ORNL/TM-7033 (May 1980).
14. W. A. Rhoades and W. W. Engle, Jr., "A New Weighted Difference Formulation for Discrete Ordinates Calculations," *Trans. Am. Nucl. Soc.* 27, 776 (November 1977).
15. W. A. Rhoades and E. T. Tomlinson, "A Comparison of Codes for Deep Penetration Transport Studies," *Proc. Computational Methods in Nucl. Eng.* 3, 4-19 to 4-33 (April 1979).
16. E. T. Tomlinson and W. A. Rhoades, "Evaluation of Flux Extrapolation Models for Discrete Ordinates Shielding Analysis," *Trans. Am. Nucl. Soc.* 34, 650-651 (June 1980).
17. K. D. Lathrop, "User's Guide for the TWOTRAN (X,Y) Program," LA-4058 (February 1969).

18. K. J. Lathrop, "Spatial Differencing of the Transport Equation: Positivity vs. Accuracy," *J. Comput. Phys.* 4, 475 (1969).
19. P. Barbucci and F. Di Pasquantonio, "Exponential Supplementary Equations for S_N Methods: The Two Dimensional Case," *Proc. 5th Intl. Conf. Reactor Shielding*, Science Press (April 1977).
20. R. E. Alcouffe, E. W. Larsen, W. F. Miller, Jr., and B. R. Wienke, "Computational Efficiency of Numerical Methods for the Multigroup, Discrete-Ordinates Neutron Transport Equation: The Slab Case," *Nucl. Sci. Eng.* 71, 111 (1979).
21. B. G. Carlson, "A Comparison of Diamond and Modified Diamond Differencing in (X,Y) Geometry," in "Transport and Reactor Theory," LA-8380-PR (May 1980).
22. J. P. Jenal, P. J. Erickson, W. A. Rhoades, D. B. Simpson, and M. L. Williams, "DOQDP, The Generation of a Computer Library for Discrete Ordinates Quadrature Sets, Part I: Fully Symmetric Quadratures and the DOQDP Computer Code, Part II: Common Quadrature Library," ORNL/TM-6023 (September 1977).
23. M. B. Emmett and W. A. Rhoades, "Monte Carlo Verification of DOT Cylindrical Boundary Condition," (to be published).
24. T. B. Fowler, D. R. Vondy, G. W. Cunningham, "Nuclear Reactor Core Analysis Code: Citation," ORNL/TM-2496, Rev. 2 (July 1969).
25. W. W. Engle, Jr. and F. R. Mynatt, "A Comparison of Two Methods of Inner Iteration Convergence Acceleration in Discrete Ordinates Codes," *Trans. Am. Nucl. Soc.* 11, 193-194 (June 1968).
26. W. A. Rhoades and E. T. Tomlinson, "Effectiveness of Three Rebalance Methods in Deep Penetration Problems," *Trans. Am. Nucl. Soc.* 33, 717-719 (November 1979).
27. H. J. Kopp, "Synthetic Method Solution of the Transport Equation," *Nucl. Sci. Eng.* 17, 65-74 (1963).
28. E. M. Gelbard and L. A. Hageman, "The Synthetic Method as Applied to the S_N Equations," *Nucl. Sci. Eng.* 37, 288-296 (1969).
29. W. A. Rhoades, "Improvements in Discrete-Ordinates Acceleration," *Trans. Am. Nucl. Soc.* (November 1981).
30. W. H. Reed, "The Effectiveness of Acceleration Techniques for Iterative Methods in Transport Theory," *Nucl. Sci. Eng.* 45, 245-254 (1971).
31. W. A. Rhoades, R. L. Childs, and W. W. Engle, Jr., "Comparison of Rebalance Stabilization Methods for Two-Dimensional Transport Calculations," *Trans. Am. Nucl. Soc.* 30, 583 (November 1978).
32. R. W. Alcouffe, "Diffusion Synthetic Acceleration Methods for the Diamond-Differenced Discrete-Ordinates Equations," *Nucl. Sci. Eng.* 64, 344-355 (1977).
33. J. E. Aull, W. A. Rhoades, and H. L. Dodds, "A Modified Approach to Diffusion Acceleration in Neutron Transport Problems," *Trans. Am. Nucl. Soc.* 32, 306-307 (June 1979).

34. R. K. Disney, Westinghouse Advanced Power Systems Division, Madison, PA (Personal Communication, 1975).
35. J. T. Ward, Jr., "An Improved Multigroup Representation of Differential Compton Scattering Cross Sections," *Trans. Am. Nucl. Soc.* **21**, 529 (June 1975).
36. M. B. Emmett, R. L. Childs, and W. A. Rhoades, "A Repair for Scattering Expansion Truncation Errors in Transport Calculations," *Trans. Am. Nucl. Soc.* **33**, 719-721 (November 1979).
37. R. Protsik and E. G. Leff, "User's Manual for DOT2DB: A Two-Dimensional Multigroup Discrete-Ordinates Transport/Diffusion Code with Anisotropic Scattering," GEAP-13537 (1969).
38. ANS Standard "Recommended Programming Practices to Facilitate the Interchange of Digital Computer Programs," prepared by Subcommittee 10, ANS Standards Committee (April 1971).
39. B. M. Carmichael, "Standard Interface Files and Procedures for Reactor Physics Codes, Version III," LA-5486-MS (February 1974).
40. R. Douglas O'Dell, "Standard Interface Files and Procedures for Reactor Physics Codes Version IV," LA-6941-MS (September 1977).
41. M. B. Emmett and W. A. Rhoades, "CARP, A Computer Code and Albedo Data Library for Use by BREESE, the MORSE Albedo Package," ORNL/TM-6503 (October 1978).
42. W. A. Rhoades, "The FBSAM Data Transmission Package for IBM 360/370 Computers," ORNL/TM-5199 (January 1976).
43. W. W. Engle, Jr., M. A. Boling, and B. W. Colston, "DTF-II, A One-Dimensional, Multigroup Neutron Transport Program," NAA-SR-10951 (March 1966).

The FIDO input method is especially devised to allow the entering or modifying of large data arrays with minimum effort. Special advantage is taken of patterns of repetition or symmetry wherever possible. The FIDO system was patterned after the input method used with the FLOCO coding system at Los Alamos, and was first applied by Atomics International to the DTF-II⁴³ code. Since that time, numerous features requested by users have been added, a free-field option has been developed, and the application of FIDO has spread to innumerable codes.

The data are entered in units called "arrays." An array comprises a group of contiguous storage locations which are to be filled with data at one time. These arrays usually correspond on a one-to-one basis with FORTRAN arrays used in the program. A group of one or more arrays read with a single call to the FIDO package comprises a "block." A special delimiter is required to signify the end of each block. Arrays within a block may be read in any order with respect to each other, but an array belonging to one block must not be shifted to another. The same array can be entered repeatedly within the same block. For example, an array can be filled with "O" using a special option, and then a few scattered locations could be changed by reading in a new set of data for that array. Some arrays can be omitted if no entries to that array are required. If no entries to any of the arrays in a block are required, but the condition requiring the block is met, the delimiter alone satisfies the input requirement.

Three major types of input are available: fixed-field input, free-field input, and user-field input. Each field includes up to three subfields. Each array is identified by a "array originator field" having two subfields:

Subfield 1:	An integer <i>array identifier</i> signals the identity of the array to follow.
Subfield 2:	An <i>array-type indicator</i> , used as follows:
	"\$" fixed-field integer array
	"*" fixed-field real array
	"\$\$" free-field integer array
	"**" free-field real array
	"U" user-field, format to be read in
	"V" user-field, previous format to be used

These subfields are written together, without blanks. Data are then placed in successive *data fields* until the required number of entries has been accounted for.

In entering data, it is convenient to think of a "index" or "pointer" which is under control of the user, and which specifies the position in the array into which the next data entry is to go. The pointer is always positioned at the first array location by entering the array originator field. The pointer subsequently moves according to the data operator chosen.

Free-field input allows an arbitrary number of fields to be entered in columns 1-72, with fields separated by one or more blanks. A field must begin and end in the same record. In general, a data field has up to three fields:

- Subfield 1: The *data numerator*, N_1 , an integer.
- Subfield 2: The *data operator*, N_2 , a character.
- Subfield 3: The *data entry*, N_3 , an integer or real number.

If subfields 1 and 2 are both used in a field, they must not be separated by blanks. Blanks may always precede subfield 3, however. All entries following a "/" in a given record are ignored.

Single data entries use only N_3 , which may be entered with or without a decimal point. The type (integer/real) will be determined by the previous array identifier:

1\$\$	1 2 3	/enter integers 1,2,3
2**	1. 1.1 2	/enter reals 1.0,1.1,2.0

An exponential field may be added, with or without the "E" identifier. No imbedded blanks within the subfield are allowed. The subfield may have no more than 9 columns, not including the exponent field. The pointer is advanced by 1 for each entry:

	2**	1.E0 .11+1 200-2	/same entries as above
	3**	1234.5678E-3	/8 columns + decimal
BUT NOT	3**	1.23456789	/too many columns

Multiple data entries use N_2 and (except "F") N_1 together with N_3 to enter several items with a single field. The type of operation is indicated by N_2 .

"K" indicates that data entry N_3 is to be repeated N_1 times. The pointer is advanced by N_1 .

"I" indicates linear interpolation. The data numerator, N_1 , indicates the number of interpolated points to be supplied. The data entry N_3 is entered, followed by N_1 interpolated entries equally spaced between that value and the data entry found in the third subfield of the next field. The next field may be a single- or multiple-data entry. The pointer is advanced by $N_1 + 1$. The field following an "I" entry is then processed normally, according to its own data operator. The "I" entry is especially valuable for specifying a spatial mesh. In "\$" arrays, interpolated values will be rounded to the nearest integer.

"L" indicates logarithmic interpolation. The effect is the same as that of "I" except that the resulting data are evenly separated in log-space. This is limited to positive real numbers, for obvious reasons.

"F" fills the remainder of the array with N_3 . N_1 is not used.

For example, the following are equivalent ways of filling an 8-entry array:

2\$\$	1 2 3 4 6 6 8 8	
2\$\$	211 4 2R6 F8	/same as above

Sequence data entries allow entries to be patterned after data entered by previous fields or previously existing in storage. Fields N_1 and N_2 , and N_3 are used, except that if N_1 is omitted, it is taken to be 1.

"Q" is used to repeat sequences of numbers without modification. The length of the sequence is given by the third subfield, N_3 . The sequence of N_3 previous entries is to be repeated N_1 times. The pointer is advanced by $N_1 * N_3$. This feature is especially valuable geometry specification.

"G" has the same effect as "Q," except that the sign of each entry of the sequence is changed each time it is entered.

"N" has the same effect as "Q," except that the order of the sequence is reversed each time it is entered.

"M" has the same effect as "N," except that the sign of the sequence is reversed each time it is entered. Options Q, G, and M are valuable in entering directional quadrature sets.

As examples:

81**	1 2 3 IQ3	/1,2,3,1,2,3
82**	-3 -2 -1 M3	/-3,-2,-1,1,2,3
83**	3R-2 G3	/-2,-2,-2,2,2,2

Zero data entries use subfields N_1 and N_2 .

"Z" sets the next N_1 entries to 0. The pointer is advanced by N_1 .

As an example:

ISS 3Z 1 2 /0,0,0,1,2

Pointer-movement data entries move the pointer without changing the data array. Subfields N_1 , N_2 or neither may be required.

"S" indicates that the pointer is to skip forward over N_1 positions, leaving those array positions unchanged.

"B" moves the pointer backward N_1 positions.

"A" moves the pointer to the position N_2 .

"E" skips over the remainder of the array. The array length criterion is satisfied by an "E," unless too many entries have been specified. No more entries to a array may be given following an "E," except that data entry may be restarted with an "A."

For example, given the following sequence of entries, comments indicate the result of reading an array of length 8:

ISS	1 2 3 4 5 6 E	/E terminates the array, leaving items 7 and 8 unchanged.
ISS	2 IS 4 A7 7 8	/now we have 2,2,4,4,5,6,7,8
ISS	2I 1 4 IB F8	/1,2,3,8,8,8,8,8

Edit fields control printing within the FIDO subroutines.

"C" causes the position of the last array item entered to be printed. This is the position of the pointer, less 1. The pointer is not moved.

"O" causes the print trigger to be turned on. The trigger is originally off. When the trigger is on, each card image is listed as it is read.

"P" causes the print trigger to be turned off.

"/" occurring in column 1 causes the entire record to be ignored as input, but to be printed as a comment in the output stream.

A *block termination field* consists of a field having only **"T"** in the second subfield.

The *reading of data to an array* is terminated when a new array originator field is supplied, or when the block is terminated by a block termination field. If a incorrect number of positions has been filled, an error edit is given, and a flag is set which will later abort execution of the problem. FIDO then continues with the next array if an array originator was read. The new array originator need not begin a new record. Otherwise, FIDO returns control to the calling program. For example:

```
ISS 1 2 3      2** F0 T
```

User-field input allows the user to specify the input format. If **"U"** is specified as the array-type indicator, the FORTRAN format to be used must be supplied in columns 1-72 of the next record. The format must be enclosed by the usual parentheses. The data for the entire array must follow on successive records. The rules of ordinary FORTRAN input as to exponents, blanks, etc., apply. If the array data do not fill the last record, the remainder of the record must be left blank. The user must insure that his format specifies the correct type of data, i.e., real or integer.

"V" has the same effect as **"U"** except that the format read in the last preceding **"U"** array is used.

For example, for an array of 4 entries:

```
10U
(6I2)
1 2 3 4
11V
4 3 2 1
```

would enter integers 1,2,3,4 into the 10th array and 4,3,2,1 into the 11th.

Fixed-field input uses 1 to 6 fields per input record, with fixed, 12 column fields. It is thoroughly described in Reference 1. Because it has been almost entirely replaced by the free-field format, its description is not repeated here.

C		-VSOR0010
		VSOR0020
C	*****	VSOR0030
C	REVISED 10 NOV 76	-VSOR0040
C		-VSOR0050
CF	VARSOR	-VSOR0060
CE	VARIABLE MESH SOURCE MOMENT DATA	-VSOR0070
C		-VSOR0080
C	*****	VSOR0090
		VSOR0100
CD	ORDER OF GROUPS IS BY DECREASING ENERGY	VSOR0110
CD	I IS THE FIRST-DIMENSION INDEX	VSOR0120
CD	J IS THE SECOND-DIMENSION INDEX	VSOR0130
CD	JM=1 FOR 1-DIMENSIONAL GEOMETRY	VSOR0140
CD	IF ISOP.GT.0, SOURCE IS FIRST-COLLISION TYPE	VSOR0150
		VSOR0160
C	-----	VSOR0170
CS	FILE STRUCTURE	-VSOR0180
CS		-VSOR0190
CS	RECORD TYPE	PRESENT IF
CS	-----	-----
CS	FILE IDENTIFICATION	ALWAYS
CS	FILE LABEL	ALWAYS
CS	FILE CONTROL	ALWAYS
CS	FILE INTEGER PARAMETERS	ALWAYS
CS		-VSOR0250
CS		-VSOR0260
CS	***** (REPEAT OVER ALL GROUPS)	-VSOR0270
CS	* SOURCE MOMENTS	ALWAYS
CS	*****	-VSOR0280
		-VSOR0290
C		-VSOR0300
CS	***** (REPEAT OVER ALL GROUPS)	-VSOR0310
CS	* SCALAR UNCOULDED FLUX	ISOP.GT.0
CS	*****	-VSOR0320
		-VSOR0330
C		-VSOR0340
C	-----	VSOR0350
		VSOR0360
		VSOR0370
C	-----	-VSOR0380
CR	FILE IDENTIFICATION	-VSOR0390
C		-VSOR0400
CL	HNAME, (HUSE(I), I=1,2), IVERS	-VSOR0410
C		-VSOR0420
CW	1+3*MULT*NUMBER OF WORDS	-VSOR0430
C		-VSOR0440
CD	HNAME HOLLERITH FILE NAME - VARSOR - (A6)	-VSOR0450
CD	HUSE(I) HOLLERITH USER IDENTIFICATION - (A6)	-VSOR0460
CD	IVERS FILE VERSION NUMBER	-VSOR0470
CD	MULT DOUBLE PRECISION PARAMETER	-VSOR0480
CD	1- A6 WORD IS SINGLE WORD	-VSOR0490
CD	2- A6 WORD IS DOUBLE PRECISION WORD	-VSOR0500
C		-VSOR0510
C	-----	VSOR0520
		VSOR0530
		VSOR0540

B-

C-----			-VSOR0550
C			-VSOR0560
CR	FILE LABEL		-VSOR0570
C			-VSOR0580
CL	DATE,USER,CHARGE,CASE,TIME,(TITL(I),I=1,12)		-VSOPC590
C			-VSOR0600
CW	17*MULT=NUMBER OF WORDS		-VSOR0610
C			-VSOR0620
CD	DATE	AS PROVIDED BY TIMER OPTION 4 - (A6)	-VSOR0630
CD	USER	AS PROVIDED BY TIMER OPTION 5 - (A6)	-VSOR0640
CD	CHARGE	AS PROVIDED BY TIMER OPTION 6 - (A6)	-VSOR0650
CD	CASE	AS PROVIDED BY TIMER OPTION 7 - (A6)	-VSOR0660
CD	TIME	AS PROVIDED BY TIMER OPTION 8 - (A6)	-VSOR0670
CD	TITL(I)	TITLE PROVIDED BY USER - (A6)	-VSOR0680
C			-VSOR0690
C-----			-VSOR0700
			VSOR0710
			VSOR0720
C-----			-VSOR0730
CR	FILE CONTROL		-VSOR0740
C			-VSOR0750
CD	IGM,NEUT,JM,LM,IMA,MMA,ISM,IMSISM,ISOP,(IDUM(N),N=1,16)		-VSOR0760
C			-VSOR0770
CW	25=NUMBER OF WORDS		-VSOR0780
C			-VSOR0790
CD	IGM	NUMBER OF ENERGY GROUPS	-VSOR0800
CD	NEUT	LAST NEUTRON GROUP	-VSOP7810
CD		(IGM IF ALL NEUTRONS, 0 IF ALL GAMMAS)	-VSOR0820
CD	JM	NUMBER OF SECOND-DIMENSION (J) INTERVALS	-VSOR0830
CD	LM	MAXIMUM LENGTH OF MOMENT EXPANSION	-VSOR0840
CD	IMA	MAXIMUM NUMBER OF FIRST-DIMENSION INTERVALS	-VSOR0850
CD	MMA	NUMBER OF BOUNDARY DIRECTIONS	-VSOR0860
CD	ISM	NUMBER OF I-BOUNDARY SETS	-VSOR0870
CD	IMSISM	TOTAL NUMBER OF I-INTERVALS, ALL I-SETS	-VSOR0880
CD	ISOP	UNCOLLIDED FLUX FLAG	-VSOR0890
CD		0 - NO UNCOLLIDED FLUX RECORDS PRESENT	-VSOR0900
CD		1 - UNCOLLIDED FLUX RECORDS PRESENT	-VSOR0910
CD	IDUM(I)	ARRAY SET TO 0	-VSOR0920
C			-VSOR0930
C-----			-VSOR0940
			VSOR0950
			VSOR0960
C-----			-VSOR0970
CR	FILE INTEGER PARAMETERS		-VSOR0980
C			-VSOR0990
CL	(LMBIG(IG),IG=1,IGM),		-VSOR1000
CL	*(IMBIS(IS),IS=1,ISM),(ISET(J),J=1,JM)		-VSOR1010
C			-VSOR1020
CW	IGM+ISM+JM=NUMBER OF WORDS		-VSOR1030
C			-VSOR1040
CD	IMBIG(IG)	LENGTH OF MOMENT EXPANSION FOR GROUP IG	-VSOR1050
CD	IMBIS(IS)	NUMBER OF INTERVALS IN ISET IS	-VSOR1060
CD	ISET(J)	I-SET ASSIGNED TO INTERVAL J	-VSOR1070
C			-VSOR1080
C-----			-VSOR1090

		VSOR1100
		VSOR1110
C-----		VSOR1120
CR	SOURCE MOMENTS	-VSOR1130
C		-VSOR1140
CL	((SORM(I,L),I=1,IMS),L=1,LMS)	-VSOR1150
C		-VSOR1160
CW	IMS*LMS=NUMBER OF WORDS	-VSOR1170
C		-VSOR1180
C	DO 1 J=1,JM	-VSOR1190
C	1 READ(N) *LIST AS ABOVE*	-VSOR1200
C		-VSOR1210
CD	SORM SOURCE BY INTERVAL AND MOMENT INDEX	-VSOR1220
CD	IMS IMBIS(IS) FOR IS CORRESPONDING TO J	-VSOR1230
CD	LMS LMBIG(IG)	-VSOR1240
C		-VSOR1250
C-----		VSOR1260
		VSOR1270
		VSOR1280
C-----		VSOR1290
CR	SCALAR UNCOLLIDED FLUX	-VSOR1300
C		-VSOR1310
CL	(FLUM(I),I=1,IMS)	-VSOR1320
C		-VSOR1330
CW	IMS=NUMBER OF WORDS	-VSOR1340
C		-VSOR1350
C	DO 1 J=1,JM	-VSOR1360
C	1 READ(N) *LIST AS ABOVE*	-VSOR1370
C		-VSOR1380
CD	FLUX UNCOLLIDED FLUX BY INTERVAL	-VSOR1390
C		-VSOR1400
C-----		VSOR1410
		VSOR1420
		VSOR1430
		VSOR1440
		VSOR1450
		VSOR1460
C	END	-VRFL0010
		VRFL0020
C*****		VRFL0030
C	REVISED 10 NOV 76	-VRFL0040
C		-VRFL0050
CF	VARFLM	-VRFL0060
CE	VARIABLE MESH FLUX MOMENT DATA WITH BOUNDARY FLUXES	-VRFL0070
C		-VRFL0080
C*****		VRFL0090
		VRFL0100
CD	ORDER OF GROUPS IS BY DECREASING ENERGY	VRFL0110
CD	I IS THE FIRST-DIMENSION INDEX	VRFL0120
CD	J IS THE SECOND-DIMENSION INDEX	VRFL0130
CD	JM=1 FOR 1-DIMENSIONAL GEOMETRY	VRFL0140
		VRFL0150

B-6

C-----			VRFL016C
CS	FILE STRUCTURE		-VRFL0170
CS			-VRFL0180
CS	RECORD TYPE	PRESENT IF	-VRFL0190
CS	-----	-----	-VRFL0200
CS	FILE IDENTIFICATION	ALWAYS	-VRFL0210
CS	FILE LABEL	ALWAYS	-VRFL0220
CS	FILE CONTIN L	ALWAYS	-VRFL0230
CS	FILE INTEGER PARAMETERS	ALWAYS	-VRFL0240
CS	FILE REAL PARAMETERS	ALWAYS	-VRFL0250
CS			-VRFL0270
CS	***** (REPEAT OVER ALL GROUPS)		-VRFL0280
CS	* FLUX MOMENTS	ALWAYS	-VRFL0290
CS	* BOUNDARY DIRECTIONAL FLUX	ALWAYS	-VRFL0300
CS	*****		-VRFL0310
C			-VRFL0320
C-----			VRFL0330
			VRFL0340
			VRFLC350
C-----			VRFL0360
CR	FILE IDENTIFICATION		-VRFL0370
C			-VRFL0380
CL	HNAME, (HUSE(I), I=1,2), IVERS		-VRFL0390
C			-VRFL0400
CW	1+3*MULT=NUMBER OF WORDS		-VRFL0410
C			-VRFL0420
CD	HNAME	HOLLERITH FILE NAME - VARFLM - (A6)	-VRFL0430
CD	HUSE(I)	HOLLERITH USER IDENTIFICATION - (A6)	-VRFL0440
CD	IVERS	FILE VERSION NUMBER	-VRFL0450
CD	MULT	DOUBLE PRECISION PARAMETER	-VRFL0460
CD		1- A6 WORD IS SINGLE WORD	-VRFL0470
CD		2- A6 WORD IS DOUBLE PRECISION WORD	-VRFL0480
C			-VRFL0490
C-----			VRFL0500
			VRFL0510
			VRFL0520
C-----			VRFL0530
C			-VRFL0540
CR	FILE LABEL		-VRFL0550
C			-VRFL0560
CL	DATE, USER, CHARGE, CASE, TIME, (TITL(I), I=1,12)		-VRFL0570
C			-VRFL0580
CW	17*MULT=NUMBER OF WORDS		-VRFL0590
C			-VRFL0600
CD	DATE	AS PROVIDED BY TIMER OPTION 4 - (A6)	-VRFL0610
CD	USER	AS PROVIDED BY TIMER OPTION 5 - (A6)	-VRFL0620
CD	CHARGE	AS PROVIDED BY TIMER OPTION 6 - (A6)	-VRFL0630
CD	CASE	AS PROVIDED BY TIMER OPTION 7 - (A6)	-VRFL0640
CD	TIME	AS PROVIDED BY TIMER OPTION 8 - (A6)	-VRFL0650
CD	TITL(I)	TITLE PROVIDED BY USER - (A6)	-VRFL0660
C			-VRFL0670
C-----			VRFL0680
			VRFL0690
			VRFL0700

C			VRFL0710
CR	FILE CONTROL		VRFL0720
C			VRFL0730
CD	IGM,NEUT,JM,LM,IMA,MMA,ISM,IMSISM,ISBT,ITER,(IDUM(N),N=1,15)		VRFL0740
C			VRFL0750
CW	25=NUMBER OF WORDS		VRFL0760
C			VRFL0770
CD	IGM	NUMBER OF ENERGY GROUPS	VRFL0780
CD	NEUT	LAST NEUTRON GROUP	VRFL0790
CD		(IGM IF ALL NEUTRONS, 0 IF ALL GAMMAS)	VRFL0800
CD	JM	NUMBER OF SECOND-DIMENSION (J) INTERVALS	VRFL0810
CD	LM	MAXIMUM LENGTH OF MOMENT EXPANSION	VRFL0820
CD	IMA	MAXIMUM NUMBER OF FIRST-DIMENSION INTERVALS	VRFL0830
CD	MMA	NUMBER OF BOUNDARY DIRECTIONS	VRFL0840
CD	ISM	NUMBER OF I-BOUNDARY SETS	VRFL0850
CD	IMSISM	TOTAL NUMBER OF I-INTERVALS, ALL I-SETS	VRFL0860
CD	ISBT	I-SET FOR SYSTEM BOUNDARIES	VRFL0870
CD	ITER	OUTER ITERATION NUMBER AT WHICH FLUX WAS WRITTEN	VRFL0880
CD	IDUM(I)	ARRAY SET TO 0	VRFL0890
C			VRFL0900
C			VRFL0910
C			VRFL0920
			VRFL0930
			VRFL0940
C			VRFL0950
CR	FILE INTEGER PARAMETERS		VRFL0960
C			VRFL0970
CL	(LMBIG(IG),IG=1,IGM),		VRFL0980
CL	*(IMBIS(IS),IS=1,ISM),(ISET(J),J=1,JM)		VRFL0990
C			VRFL1000
CW	IGM+ISM+JM=NUMBER OF WORDS		VRFL1010
C			VRFL1020
CD	LMBIG(IG)	LENGTH OF MOMENT EXPANSION FOR GROUP IG	VRFL1030
CD	IMBIS(IS)	NUMBER OF INTERVALS IN ISET IS	VRFL1040
CD	ISET(J)	I-SET ASSIGNED TO INTERVAL J	VRFL1050
C			VRFL1060
C			VRFL1070
			VRFL1080
			VRFL1090
C			VRFL1100
CR	FILE REAL PARAMETERS		VRFL1110
C			VRFL1120
CL	(Z(J),J=1,JM1),((R(I,IS),I=1,IM1),IS=1,ISM),		VRFL1130
CL	*(ENER(IG),IG=1,IGM),EMIN,ENEUT,EV,DEVOK,EPPK,POWER,		VRFL1140
CL	*(DUMRL(I),I=1,13)		VRFL1150
C			VRFL1160
CW	JM+IMSISM+ISM+IGM+20=NUMBER OF WORDS		VRFL1170
C			VRFL1180
CD	Z(J)	J-INTERVAL BOUNDARIES	VRFL1190
CD	R(I,IS)	I-INTERVAL BOUNDARIES FOR I-SET I	VRFL1200
CD	ENER(IG)	TOP ENERGY BOUNDARY OF GROUP IG	VRFL1210
CD	EMIN	BOTTOM ENERGY BOUNDARY OF GROUP IGM	VRFL1220
CD	ENEUT	BOTTOM ENERGY BOUNDARY OF GROUP NEUT	VRFL1230
CD		(0 IF NEUT=0)	VRFL1240

B-8

CD	EV	EIGENVALUE	-VRFL1250
CD	DEVOK	DERIVATIVE OF EV VS. EPPK	-VRFL1260
CD	EPPK	EFFECTIVE MULTIPLICATION FACTOR	-VRFL1270
CD	POWER	POWER (WATTS) TO WHICH FLUX IS NORMALIZED	-VRFL1280
CD	DUMRL	ARRAY SET TO 0	-VRFL1290
CD	JM1	JM+1	-VRFL1300
CD	IM1	IMBIS(IS)+1	-VRFL1310
C			-VRFL1320
C			-VRFL1330
			VRFL1340
			VRFL1350
			VRFL1360
			VRFL1370
C			-VRFL138C
CR	FLUX MOMENTS		-VRFL1390
C			-VRFL1400
CL	(FLUM(I,L),I=1,LMS),L=1,LMS)		-VRFL1410
C			-VRFL1420
CW	LMS*LMS=NUMBER OF WORDS		-VRFL1430
C			-VRFL1440
C	DO 1 J=1,JM		-VRFL1450
C	1 READ(N) *LIST AS ABOVE*		-VRFL1460
C			-VRFL1470
CD	FLUM	FLUX BY INTERVAL AND MOMENT INDEX	-VRFL1480
CD	IMS	IMBIS(IS) FOR IS CORRESPONDING TO J	-VRFL1490
CD	LMS	LMBIG(IG)	-VRFL1500
C			-VRFL1510
C			-VRFL1520
			VRFL1530
			VRFL1540
C			-VRFL1550
CR	BOUNDARY DIRECTIONAL FLUX		-VRFL1560
C			-VRFL1570
CL	((PIO(M,J),M=1,MMA),J=1,JM), ((FJO(M,I),M=1,MMA),I=1,IMB)		-VRFL1580
C			-VRFL1590
CW	MMA*(JM+IMB)=NUMBER OF WORDS		-VRFL1600
C			-VRFL1610
CD	PIO	DIRECTIONAL FLUX OUTGOING BY DIRECTION AND J-INTERVAL	-VRFL1620
CD	FJO	DIRECTIONAL FLUX OUTGOING BY DIRECTION AND I-INTERVAL. FJO=0 FOR A 1-D GEOMETRY	-VRFL1630
CD	IMB	IMBIS(IS) FOR IS CORRESPONDING TO ISBT	-VRFL1640
C			-VRFL1650
C			-VRFL1660
C			-VRFL1670
C			-VRFL1680
			VRFL1690
			VRFL1700
			VRFL1710
			VRFL1720
			VRFL1730

END

```

C -BND$0010
C*****BND$0020
C          PROPOSED  8 OCT 76 -BND$0030
C -BND$0040
C          BND$0050
C          MULTIPLE BOUNDARY SOURCE DATA -BND$0060
C -BND$0070
C*****BND$0080
C          BND$0090
CD          ORDER OF GROUPS IS BY DECREASING ENERGY BND$0100
CD          I IS THE FIRST-DIMENSION INDEX BND$0110
CD          J IS THE SECOND-DIMENSION INDEX BND$0120
CD          JM=1 FOR 1-DIMENSIONAL GEOMETRY BND$0130
CD          NINTSR=NJNTSR=1 FOR EXTERNAL BOUNDARY DATA, BND$0140
CD          AND Z(1)=R(1)=0 IN THAT CASE BND$0150
C          BND$0160
C-----BND$0170
CS          FILE STRUCTURE -BND$0180
CS -BND$0190
CS          RECORD TYPE          PRESENT IF -BND$0200
CS-----BND$0210
CS          FILE IDENTIFICATION  ALWAYS -BND$0220
CS          FILE LABEL          ALWAYS -BND$0230
CS          FILE CONTROL        ALWAYS -BND$0240
CS          MESE DESCRIPTION     ALWAYS -BND$0250
CS -BND$0260
CS          ***** (REPEAT OVER ALL GROUPS) -BND$0270
CS          *          I-BOUNDARY DIRECTIONAL SOURCE NINTSR.GT.0 -BND$0280
CS          *          J-BOUNDARY DIRECTIONAL SOURCE NJNTSR.GT.0 -BND$0290
CS          ***** -BND$0300
C -BND$0310
C-----BND$0320
C          BND$0330
C          BND$0340
C-----BND$0350
CR          FILE IDENTIFICATION -BND$0360
C -BND$0370
CL          HNAME, (HUSE(I), I=1,2), IVERS -BND$0380
C -BND$0390
CW          1+3*MULT=NUMBER OF WORDS -BND$0400
C -BND$0410
CD          HNAME          HOLLERITH FILE NAME - BND$0420
CD          HUSE(I)        HOLLERITH USER IDENTIFICATION - (A6) -BND$0430
CD          IVERS          FILE VERSION NUMBER -BND$0440
CD          MULT           DOUBLE PRECISION PARAMETER -BND$0450
CD          1- A6 WORD IS SINGLE WORD -BND$0460
CD          2- A6 WORD IS DOUBLE PRECISION WORD -BND$0470
C -BND$0480
C-----BND$0490
C          BND$0500
C          BND$0510

```

B-10

C	-----		-BND50520
CR	FILE LABEL		-BND50530
C			-BND50540
CL	DATE,USER,CHARGE,CASE,TIME, (TITL(I),I=1,12)		-BND50550
C			-BND50560
CW	17*MULT=NUMBER OF WORDS		-BND50570
C			-BND50580
CD	DATE	AS PROVIDED BY TIMER OPTION 4 - (A6)	-BND50590
CD	USER	AS PROVIDED BY TIMER OPTION 5 - (A6)	-BND50600
CD	CHARGE	AS PROVIDED BY TIMER OPTION 6 - (A6)	-BND50610
CD	CASE	AS PROVIDED BY TIMER OPTION 7 - (A6)	-BND50620
CD	TIME	AS PROVIDED BY TIMER OPTION 8 - (A6)	-BND50630
CD	TITL(I)	TITLE PROVIDED BY USER - (A6)	-BND50640
C			-BND50650
C	-----		-BND50660
			BND50670
			BND50680
C	-----		-BND50690
CR	FILE CONTROL		-BND50700
C			-BND50710
CL	IGM,JM,IMA,MMA,NINTSR,NJNTSR, (IDUM(I),I=1,19)		-BND50720
C			-BND50730
CW	25=NUMBER OF WORDS		-BND50740
C			-BND50750
CD	IGM	NUMBER OF ENERGY GROUPS	-BND50760
CD	JM	NUMBER OF SECOND-DIMENSION (J) INTERVALS	-BND50770
CD	IMA	NUMBER OF FIRST-DIMENSION INTERVALS	-BND50780
CD	MMA	NUMBER OF BOUNDARY DIRECTIONS	-BND50790
CD	NINTSR	NUMBER OF I-BOUNDARY SOURCES	-BND50800
CD	NJNTSR	NUMBER OF J-BOUNDARY SOURCES	-BND50810
CD	IDUM(I)	ARRAY SET TO 0	-BND50820
C			-BND50830
C	-----		-BND50840
			BND50850
			BND50860
C	-----		-BND50870
CR	MESH DESCRIPTION		-BND50880
C			-BND50890
CL	(Z(J),J=1,NJNTSR),(R(I),I=1,NINTSR)		-BND50900
C			-BND50910
CW	JM+IMA=NUMBER OF WORDS		-BND50920
C			-BND50930
CD	Z(J)	POSITION FROM WHICH J-BOUNDARY SOURCES TAKEN	-BND50940
CD	R(I)	POSITION FROM WHICH I-BOUNDARY SOURCES TAKEN	-BND50950
C			-BND50960
C	-----		-BND50970
			BND50980
			BND50990

B-11

```

C-----BNDS1000
CR      I-BOUNDARY DIRECTIONAL SOURCES      -BNDS1010
C-----BNDS1020
CL      ((BIJ(M,J,N),M=1,MMA),J=1,JM),N=1,NINTSR) -BNDS1030
C-----BNDS1040
CW      MMA*JM*NINTSR=NUMBER OF WORDS      -BNDS1050
C-----BNDS1060
CD      BIJ(M,J,N)      SOURCE IN DIRECTION M, INTERVAL J, SET N -BNDS1070
C-----BNDS1080
C-----BNDS1090
BNDS1100
BNDS1110
C-----BNDS1120
CR      J-BOUNDARY DIRECTIONAL SOURCES      -BNDS1130
C-----BNDS1140
CL      (((BJI(M,I,N),M=1,MMA),I=1,IMA),N=1,NJNTR) -BNDS1150
C-----BNDS1160
CW      MMA*IMA*NJNTR=NUMBER OF WORDS      -BNDS1170
C-----BNDS1180
CD      BJI(M,I,N)      SOURCE IN DIRECTION M, INTERVAL I, SET N -BNDS1190
C-----BNDS1200
C-----BNDS1210
BNDS1220
BNDS1230
BNDS1240
BNDS1250
BNDS1260
C      END                                -ORDS0010
C-----ORDS0020
C*****ORDS0030
C      REVISED 10 NOV 76                -ORDS0040
C-----ORDS0050
CP      ORDOSW                          -ORDS0060
CE      WORKING FILE FOR THE OAK RIDGE DISCRETE ORDINATES SYSTEM -ORDS0070
C-----ORDS0080
C*****ORDS0090
ORDS0100
ORDS0110
ORDS0120
ORDS0130
ORDS0140
C-----ORDS0150
CS      FILE STRUCTURE                  -ORDS0160
CS-----ORDS0170
CS      RECORD TYPE                      PRESENT IF      -ORDS0180
CS-----ORDS0190
CS      FILE IDENTIFICATION              ALWAYS          -ORDS0200
CS      FILE LABEL                      ALWAYS          -ORDS0210
CS      FILE LENGTH DATA                ALWAYS          -ORDS0220
CS      FILE INTEGER PARAMETERS          ALWAYS          -ORDS0230
CS      FILE REAL PARAMETERS             ALWAYS          -ORDS0240
CS      QUADRATURE SET LENGTHS           NQUAD.GT.0       -ORDS0250

```

B-12

CS	***** (REPEAT OVER NQUAD QUADRATURE SETS)	-ORDS0260
CS	* QUADRATURE SET IDENTIFICATION NQUAD.GT.0	-ORDS0270
CS	* QUADRATURE DATA NQUAD.GT.0	-ORDS0280
CS	*****	-ORDS0290
C		-ORDS0300
CS	PRINCIPAL CROSS SECTION DESCRIPTN MTM.GT.0	-ORDS0310
CS	NUCLID2 IDENTIFICATION MTM.GT.0	-ORDS0320
C		-ORDS0330
CS	***** (REPEAT OVER IGM ENERGY GROUPS)	-ORDS0340
CS	* CROSS SECTION DATA MTM.GT.0	-ORDS0350
CS	*****	-ORDS0360
CS		-ORDS0370
C		-ORDS0380
		ORDS0390
		ORDS0400
C		-ORDS0410
CR	FILE IDENTIFICATION	-ORDS0420
C		-ORDS0430
CL	HNAME, (HUSE(I), I=1,2), IVERS	-ORDS0440
C		-ORDS0450
CW	NUMBER OF WORDS = 1 + 3*MULT	-ORDS0460
C		-ORDS0470
CD	HNAME HOLLERITH FILE NAME - ORDOSW - (A6)	-ORDS0480
CD	HUSE(I) HOLLERITH USER IDENTIFICATION - (A6)	-ORDS0490
CD	IVERS FILE VERSION NUMBER	-ORDS0500
CD	MULT DOUBLE PRECISION PARAMETER	-ORDS0510
CD	1- A6 WORD IS SINGLE WORD	-ORDS0520
CD	2- A6 WORD IS DOUBLE PRECISION WORD	-ORDS0530
C		-ORDS0540
C		-ORDS0550
		ORDS0560
		ORDS0570
C		-ORDS0580
CR	FILE LABEL	-ORDS0590
C		-ORDS0600
CL	DATE, USER, CHARGE, CASE, TIME, (TITL(I), I=1,12)	-ORDS0610
C		-ORDS0620
CW	NUMBER OF WORDS = MULT*17	-ORDS0630
C		-ORDS0640
CD	DATE AS PROVIDED BY TIMER OPTION 4 - (A6)	-ORDS0650
CD	USER AS PROVIDED BY TIMER OPTION 5 - (A6)	-ORDS0660
CD	CHARGE AS PROVIDED BY TIMER OPTION 6 - (A6)	-ORDS0670
CD	CASE AS PROVIDED BY TIMER OPTION 7 - (A6)	-ORDS0680
CD	TIME AS PROVIDED BY TIMER OPTION 8 - (A6)	-ORDS0690
CD	TITL(I) TITLE PROVIDED BY USER - (A6)	-ORDS0700
C		-ORDS0710
C		-ORDS0720

B-13

		ORDS0730
		ORDS0740
C		ORDS0750
CR	FILE LENGTH DATA	-ORDS0760
C		-ORDS0770
CL	NBRINT,NBRREL,NBRREC,IGM,MTM,NQUAD,NUMDM(14)	-ORDS0780
C		-ORDS0790
CW	NUMBER OF WORDS = 20	-ORDS0800
C		-ORDS0810
CD	NBRINT	NUMBER OF INTEGER PARAMETERS = 20 -ORDS0820
CD	NBRREL	NUMBER OF REAL PARAMETERS = 10 + (1+NC)*IGM -ORDS0830
CD	NBRREC	NUMBER OF FIRST RECORD OF CROSS SECTION DATA -ORDS0840
CD	IGM	NUMBER OF ENERGY GROUPS -ORDS0850
CD	MTM	NUMBER OF NUCLIDES -ORDS0860
CD	NQUAD	NUMBER OF QUADRATURE SETS -ORDS0870
CD	NUMDM(I)	ARRAY SET TO 0 -ORDS0880
C		-ORDS0890
C		ORDS0900
		ORDS0910
		ORDS0920
C		ORDS0930
CR	FILE INTEGER PARAMETERS	-ORDS0940
C		-ORDS0950
CL	IHT,IHS,IHM,NEUT,IADJ,NSCAT,NCHI,NUMIN(13)	-ORDS0960
C		-ORDS0970
CW	NUMBER OF WORDS = NBRINT	-ORDS0980
C		-ORDS0990
CD	NEUT	LAST NEUTRON GROUP -ORDS1000
CD		(IGM IF ALL NEUTRONS, 0 IF ALL GAMMAS) -ORDS1010
CD	IHT	TABLE POSITION OF TOTAL CROSS SECTION -ORDS1020
CD	IHS	TABLE POSITION OF SELF-SCATTER CROSS SECTION -ORDS1030
CD	IHM	CROSS SECTION TABLE LENGTH -ORDS1040
CD	IADJ	ADJOINT FLAG -ORDS1050
CD		0- NON-ADJOINT DATA -ORDS1060
CD		1- ADJOINT DATA -ORDS1070
CD	NSCAT	NUMBER OF LEGENDRE COMPONENTS BEYOND 0TH -ORDS1080
CD	NCHI	NUMBER OF FISSION SPECTRA -ORDS1090
CD	NUMIN(1)	ARRAY SET TO 0 -ORDS1100
C		-ORDS1110
C		ORDS1120
		ORDS1130
		ORDS1140
C		ORDS1150
CR	FILE REAL PARAMETERS	-ORDS1160
C		-ORDS1170
CL	(ENER(I),I=1 IGM),EMIN,ENEUT,((CHI(I,NC),I=1,IGM),NC=1,NCHI),	-ORDS1180
CL	* DUMRL(8)	-ORDS1190
C		-ORDS1200
CW	NUMBER OF WORDS = NBRREL	-ORDS1210
C		-ORDS1220
CD	ENER(IG)	TOP ENERGY BOUNDARY OF GROUP IG -ORDS1230
CD	EMIN	BOTTOM ENERGY BOUNDARY OF GROUP IGM -ORDS1240

B-14

CD	ENEUT	BOTTOM ENERGY BOUNDARY OF GROUP NEUT	-ORDS1250
CD		(0 IF NEUT=0)	-ORDS1260
CD	CHI(I,NC)	FRACTIONAL FISSION YIELD BY ENERGY GROUP	-ORDS1270
CD	DUMRL(I)	ARRAY SET TO 0	-ORDS1280
C			-ORDS1290
C			-ORDS1300
			ORDS1310
			ORDS1320
C			-ORDS1330
CR	QUADRATURE SET LENGTHS		-ORDS1340
C			-ORDS1350
CL	(MM(MS),MS=1,NQUAD)		-ORDS1360
C			-ORDS1370
CW	NUMBER OF WORDS = MULT*NQUAD		-ORDS1380
C			-ORDS1390
CD	MM(MS)	NUMBER OF DIRECTIONS BY QUADRATURE SET	-ORDS1400
C			-ORDS1410
C			-ORDS1420
C			-ORDS1430
			ORDS1440
			ORDS1450
C			-ORDS1460
CR	QUADRATURE SET IDENTIFICATION		-ORDS1470
C			-ORDS1480
CL	(QID(I),I=1,2)		-ORDS1490
C			-ORDS1500
CW	NUMBER OF WORDS = MT*2		-ORDS1510
C			-ORDS1520
CD	QID(I)	QUADRATURE SET LABEL - (A6)	-ORDS1530
C			-ORDS1540
C			-ORDS1550
			ORDS1560
			ORDS1570
C			-ORDS1580
CR	QUADRATURE DATA		-ORDS1590
C			-ORDS1600
CL	(W(M),I=1,MX),(EMU(M),I=1,MX),(ETA(M),I=1,MX)		-ORDS1610
C			-ORDS1620
CW	NUMBER OF WORDS = 3*MX		-ORDS1630
C			-ORDS1640
CD	W(M)	WEIGHT BY DIRECTION	-ORDS1650
CD	EMU(M)	FIRST DIRECTION COSINE BY DIRECTION	-ORDS1660
CD	ETA(M)	SECOND DIRECTION COSINE BY DIRECTION	-ORDS1670
CD	MX	MM(MS) FOR THIS QUADRATURE SET	-ORDS1680
C			-ORDS1690
C			-ORDS1700
			ORDS1710
			ORDS1720

C			-----ORDS1730
C			-----ORDS1740
CR	PRINCIPAL CROSS SECTION DESCRIPTION		-----ORDS1750
C			-----ORDS1760
CL	(PCSD(IH), IH-1, IHT)		-----ORDS1770
C			-----ORDS1780
CW	NUMBER OF WORDS = MULT*IHT		-----ORDS1790
C			-----ORDS1800
CD	PCSD(IH) PRINCIPAL CROSS SECTION LABELS - (A6)		-----ORDS1810
C			-----ORDS1820
C			-----ORDS1830
			ORDS1840
			ORDS1850
C			-----ORDS1860
CR	NUCLIDE IDENTIFICATION		-----ORDS1870
C			-----ORDS1880
CL	(NUC(MT), MT=1, MTM)		-----ORDS1890
C			-----ORDS1900
CW	NUMBER OF WORDS = MULT*MTM		-----ORDS1910
C			-----ORDS1920
CD	NUC(MT) NUCLIDE LABELS - (A6)		-----ORDS1930
C			-----ORDS1940
C			-----ORDS1950
			ORDS1960
			ORDS1970
C			-----ORDS1980
CR	CROSS SECTION DATA		-----ORDS1990
C			-----ORDS2000
CL	((SIG(IH, MT), IH=1, IHP), MT=1, MTM)		-----ORDS2010
C			-----ORDS2020
CW	NUMBER OF WORDS = IHP*MTM		-----ORDS2030
C			-----ORDS2040
CD	SIG(IH, MT) CROSS SECTION DATA BY TABLE POSITION, THEN BY		-----ORDS2050
CD		NUCLIDE. TABLE POSITIONS CONTAIN - -	-----ORDS2060
CD	1 TO IHT-5	ARBITRARY DATA, SPECIFIED BY USER, OR ABSENT	-----ORDS2070
CD	IHT-4	FISSION YIELD FRACTION (RECOMMENDED)	-----ORDS2080
CD	IHT-3	FISSION CROSS SECTION (RECOMMENDED)	-----ORDS2090
CD	IHT-2	ABSORPTION CROSS SECTION	-----ORDS2100
CD	IHT-1	NEUTRON PRODUCTION CROSS SECTION	-----ORDS2110
CD	IHT-0	TOTAL CROSS SECTION	-----ORDS2120
CD	IHT+1	UPSCATTER FROM GROUP IG+IHS-IHT-1 TO IG	-----ORDS2130
CD	.	.	-----ORDS2140
CD	.	.	-----ORDS2150
CD	IHS-1	UPSCATTER FROM GROUP IG+1 TO IG	-----ORDS2160
CD	IHS	SELF SCATTER FOR GROUP IG	-----ORDS2170
CD	IHS+1	DOWNSCATTER FROM GROUP IG-1 TO IG	-----ORDS2180
CD	.	.	-----ORDS2190
CD	.	.	-----ORDS2200
CD	IHM	DOWNSCATTER FROM GROUP IG+IHS-IHM TO IG	-----ORDS2210
CD	IHM+1	UPSCATTER FROM GROUP G TO ALL GROUPS	-----ORDS2212
CD		(PRESENT ONLY IF IHS.G". IHT+1	-----ORDS2214

B-16

CD
CD
CD
CD
CD
CD
CD
C
C
C

IHS MAY BE IHT+1 IF UPSCATTER IS ABSENT.
TRANSFERS FROM GROUPS .LT. 1 OR .GT. IGM
ARE 0. POSITIONS .LE. IHT ARE 0 FOR PL
COMPONENTS OTHER THAN OTH.
EACH COMPONENT OF A PL SET IS TREATED AS A
SEPARATE NUCLIDE. THUS, M SETS WOULD
COMPRISE M*NQUAD+M NUCLIDES
IHP IHM UNLESS IHS.GT.IHT+1, THEN IHM+1

-ORDS2215
-ORDS2220
-ORDS2230
-ORDS2240
-ORDS2250
-ORDS2260
-OPT 270
-ORDS2280
-ORDS2288
-ORDS2290
-----ORDS2300
ORDS2310
ORDS2320
ORDS2330
ORDS2340
ORDS2350

END

1. Mynatt, F. R., "A User's Manual for DOT," K-1694 unpublished.
This originated as a rough draft of a Union Carbide document, and bore no date, although it was distributed informally in or about 1968. It sufficed for both DOT and DOT II users.
2. Engle, W. W., Jr., and F. R. Mynatt, "A Comparison of Two Methods of Inner Iteration Convergence Acceleration in Discrete Ordinates Codes," *Trans. Am. Nucl. Soc.* 11, 1, 183-194 (June 1968).
This described the space-dependent scaling introduced in DOT II.
3. Engle, W. W., Jr., F. R. Mynatt, and R. S. Booth, "One-Dimensional Time-Dependent Discrete Ordinates," *Trans. Am. Nucl. Soc.* 12, 1, 400-401 (June 1969).
This was the first discussion of the weighted difference flux model used in DOT III.
4. Mynatt, F. R., F. J. Muckenthaler, and P. N. Stevens, "Development of Two-Dimensional Discrete Ordinates Transport Theory for Radiation Shielding," Union Carbide Corporation Nuclear Division, Oak Ridge, TN, CTC-INF-952 (August 1969).
This gave a thorough discussion of the mathematical background and extensive comparison with experiments.
5. Mynatt, F. R., et al., "The DOT III Two-Dimensional Discrete Ordinates Transport Code," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL/TM-4280 (September 1973).
This was a formal user's manual for DOT III. Together with a few pages of supplementary notes, it was also used for DOT 3.5.
6. Rhoades, W. A., "The DOT IV Variable Mesh Discrete Ordinates Transport Code," *Proc. 5th Intl. Conf. Reactor Shielding* (April 1977), Science Press, 724-730.
This described general features of DOT IV, and showed important advantages of the variable mesh features in solving streaming problems.
7. Jenal, J. P., et al., "The Generation of a Computer Library for Discrete Ordinates Quadrature Sets," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL/TM-6023 (October 1977).
This discussed an improved computer code for generating standard directional quadrature sets, described the methodology used in the generation, and listed a large number of standard sets.
8. Rhoades, W. A., and W. W. Engle, Jr., "A New Weighted Difference Formulation for Discrete Ordinates Calculations," *Trans. Am. Nucl. Soc.* 27, 776 (November 1977).
This discussed the failure of weighted difference to provide accurate K calculations, although it converged faster. The "theta-weighted" model used in DOT IV was introduced. The new model combined rapid convergence with accuracy in the demonstration problem.
9. Rhoades, W. A., R. L. Childs, and W. W. Engle, Jr., "Comparison of Rebalance Stabilization Methods for Two-Dimensional Transport Calculations," *Trans. Am. Nucl. Soc.* 30, 583 (November 1978).
Derivations and formulations for the stabilized rebalance used in DOT 3.5 and DOT IV were given here. The stabilization was unimportant for moderate scattering ratios, but essential for ratios near unity. For large, difficult problems, damping was increased to several times that required for stability in order to speed convergence.

10. Rhoades, W. A., et al., "The DOT IV Two-Dimensional Discrete-Ordinates Transport Code with Space-Dependent Mesh and Quadrature." Oak Ridge National Laboratory, Oak Ridge, TN, ORNL/TM-6529 (January 1979).

This was a detailed programmer's and user's manual for DOT IV.

11. Rhoades, W. A., and E. T. Tomlinson, "A comparison of Codes for Deep Penetration Transport Studies," *Proc. Computational Methods in Nucl. Eng.* 3, 4-19 to 4-33 (April 1979).

This compared DOT and TWOTRAN in speed, construction, and problem solving capability. Although TWOTRAN lacks many features needed for large transport problems, the execution speed of its simple linear-zero model was admirable. The two codes were shown to give drastically different results, and the reasons were not clearly understood at that time. DOT generally required fewer iterations to converge.

12. Aull, J. E., W. A. Rhoades, and H. L. Dodds, "A Modified Approach to Diffusion Acceleration in Neutron Transport Problems," *Trans. Am. Nucl. Soc.* 32, 306-307 (June 1979).

This described a "Consistent Diffusion Acceleration" which was fully compatible with all of the difference schemes used in DOT IV, and improved convergence on difficult problems.

13. Rhoades, W. A. and E. T. Tomlinson, "Effectiveness of Three Rebalance Methods in Deep Penetration Problems," *Trans. Am. Nucl. Soc.* 33, 717-719 (November 1979).

This discussed stabilized conventional space rebalance (CRB), consistent diffusion acceleration (CDA), and a new "gradient rebalance" (GRB) which combined certain features of both. CRB and GRB used an acceleration matrix determined from preliminary results of the problem being solved. CDA used a fixed matrix determined from cross section data. For difficult problems, CDA was better, unless the problem contained large internal voids.

14. Emmett, M. B., R. L. Childs, and W. A. Rhoades, "A Repair for Scattering Expansion Truncation Errors in Transport Calculations," *Trans. Am. Nucl. Soc.* 33, 719-721 (November 1979).

A special version of the MORSE code was used to examine the effectiveness of a negative-source fixup used in DOT IV. The fixup was seen to change obviously wrong numbers into reasonably accurate results.

15. Aull, J. E., "Acceleration of the Inner Iteration of the DOT IV Transport Code Using a New Source Correction Scheme," University of Tennessee, Knoxville, TN, M.S. Thesis (December 1979).

A thorough discussion of consistent diffusion acceleration as applied to fixed-source transport problems was given.

16. Aull, J. E., "The Stability of the Mesh-Cornered Synthetic Method of Diffusion Acceleration of the DOT IV Transport Code," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL/TM-7097 (October 1979).

This discussed the stability limits of unstabilized and stabilized conventional space-dependent rebalance, mesh cornered diffusion acceleration, and mesh centered consistent diffusion acceleration.

17. Tomlinson, E. T., W. A. Rhoades, and W. W. Engle, Jr., "Flux Extrapolation Models Used in the DOT IV Discrete Ordinates Neutron Transport Code," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL/TM-7033 (May 1980).
This completely resolved the discrepancies between DOT and TWOTRAN results. The consequences of various flux difference models, including linear-zero, linear-step, weighted, and theta-weighted were examined in depth.
18. Rhoades, W. A., "Improvements in Discrete-Ordinates Acceleration," *Trans. Am. Nucl. Soc.* (November 1981)
It was found that synthesis acceleration could be used to accelerate convergence of implicit boundary conditions. A troublesome damping parameter was made somewhat self-adjusting.
19. Emmett, M. B. and W. A. Rhoades, "Monte Carlo Verification of DOT Cylindrical Boundary Condition," (to be published).
Historically, neither reflective nor white boundary conditions have proven very satisfactory in representing arrays of fuel rods. A new "cylindrical" boundary condition was very satisfactory in this respect.