ornl

ORNL/TM-13221

# OAK RIDGE NATIONAL LABORATORY

LOCKHEED MARTIN

Computational Physics and Engineering Division

# THE TORT THREE-DIMENSIONAL DISCRETE ORDINATES NEUTRON/PHOTON TRANSPORT CODE (TORT Version 3)

W. A. Rhoades
D. B. Simpson



October 1997

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
LOCKHEED MARTIN ENERGY RESEARCH CORPORATION
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-96OR22464

ORNL/TM-13221

Computational Physics and Engineering Division

# THE TORT THREE-DIMENSIONAL DISCRETE ORDINATES NEUTRON/PHOTON TRANSPORT CODE (TORT Version 3)

W. A. Rhoades
D. B. Simpson

October 1997



Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
LOCKHEED MARTIN ENERGY RESEARCH CORPORATION
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-96OR22464

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# REGION 0. ABSTRACT AND ACKNOWLEDGEMENTS

## 0.1 ABSTRACT

TORT calculates the flux or fluence of neutrons and/or photons throughout three-dimensional systems due to particles incident upon the system's external boundaries, due to fixed internal sources, or due to sources generated by interaction with the system materials. The transport process is represented by the Boltzmann transport equation. The method of discrete ordinates is used to treat the directional variable, and a multigroup formulation treats the energy dependence. Anisotropic scattering is treated using a Legendre expansion. Various methods are used to treat spatial dependence, including nodal and characteristic procedures that have been especially adapted to resist numerical distortion. A method of body overlay assists in material zone specification, or the specification can be generated by an external code supplied by the user.

Several special features are designed to concentrate machine resources where they are most needed. The directional quadrature and Legendre expansion can vary with energy group. A discontinuous mesh capability has been shown to reduce the size of large problems by a factor of roughly three in some cases.

The emphasis in this code is a robust, adaptable application of time-tested methods, together with a few well-tested extensions.

## 0.2 ACKNOWLEDGEMENTS

W. A. Rhoades, who retired from the Oak Ridge National Laboratory at the end of 1994, was the primary author and developer of TORT. Before retiring, he provided version 2.12.14 of TORT to the Radiation Shielding Information Center for distribution along with an initial draft of this document.

R. L. Childs was one of the originators of TORT, and the nodal and characteristic solutions are largely his. He also contributed the optional Cray Assembly Language routines that boost speed on the large mainframes, and numerous other things. Other contributors to Oak Ridge methodology and codes should be noted, especially W. W. Engle, F. R. Mynatt, E. T. Tomlinson, and D. R. Vondy. Advice and counsel many years ago from K. D. Lathrop and B. G. Carlson of Los Alamos National Laboratory is still remembered and appreciated. More recently, W. W. Walters, also of Los Alamos, contributed important assistance to our early work with the nodal methods.

Valuable help with the manuscript preparation was provided by Ruth Lawson and Angie Alford of Oak Ridge.

# REGION 1. PROGRAM ABSTRACT

## 1.1 PROGRAM NAME AND TITLE

TORT – Three Dimensional Oak Ridge Discrete Ordinates Neutron/Photon Transport Code

## 1.2 COMPUTER FOR WHICH PROGRAM IS DESIGNED AND OTHER MACHINE VERSIONS AVAILABLE

The standard Oak Ridge releases of TORT are designed to operate efficiently on Cray mainframes and on IBM, DEC, HP, and SUN workstations. Installation instructions for these applications are provided with the code. Adaptations to Silicon Graphics workstations and to PC's by users have been reported.

## 1.3 PROBLEM SOLVED

TORT calculates the flux or fluence throughout a two- or three-dimensional geometric system due to particles incident upon the system from extraneous sources or generated internally as a result of particle interaction with the system materials. The principal application is to the deep penetration of neutrons and photons. Reactor eigenvalue problems can also be solved. Numerous printed edits of the results are available, and many results can be transferred to output files for subsequent analysis.

## 1.4 METHOD OF SOLUTION

The Boltzmann transport equation is solved using the method of discrete ordinates to treat the directional variable and the weighted difference, nodal, or characteristic method to treat spatial variables. Energy dependence is treated using a multigroup formulation. Time dependence is not allowed. Starting in one corner of a mesh, at the highest energy, and with starting guesses for implicit sources, boundary conditions and recursion relationships are used to sweep into the mesh for each discrete direction. Integral quantities such as scalar flux are obtained from weighted sums over the directional results. The calculation then proceeds to lower energy groups.

Iterations are used to resolve implicitness caused by scattering between directions within a single energy group, by scattering from one energy group to another group previously calculated, by fission, and by certain boundary conditions. Methods are available to accelerate convergence.

Fixed sources can be specified at either external or internal mesh boundaries, or distributed within mesh cells. Either cylindrical (RΘZ) or Cartesian (XYZ) geometry is supported, as well as several two-dimensional subsets.

## 1.5 RESTRICTIONS ON THE COMPLEXITY OF THE PROBLEM

Since dynamic dimensioning is used throughout, there are no artificial limits on individual variables, although the machine size obviously provides an overall limit. The code document gives a lengthy method of estimating memory and disk requirements, but an example may be better suited to this abstract. One of the demonstration problems, a large concrete building model having 104,247 mesh cells, 140 directions, P-1 scattering expansion, and 69 energy groups was solved in less than 1 Megaword of memory. The largest file, the flux moment storage, was about 29 Megawords. Boundary fluxes used about 1.1 Megawords. A similar problem with about 193,200 mesh cells was solved in about 1.2 Megawords of memory. Problems with 3 million mesh cells are now solved on Cray mainframes, and larger problems are being planned.

## 1.6 TYPICAL RUNNING TIME

The running time varies from a few seconds to many hours, depending upon the complexity of the problem. For large problems, the time for flux solution dominates the overall cpu usage, and the rate is:

$$\frac{\text{mesh cells} * \text{directions} * \text{energy groups} * \text{iterations per group}}{\text{flux calculation time (fct)}}$$

The test problem #6 distributed with the code uses 104,247 mesh cells with 60 discrete directions. It is able to provide an iteration over all mesh cells and all directions for one energy group, including all overhead activities, in the following times, expressed in minutes:

| | Cray Y-MP | | IBM RS/6000 Model 320 | |
|---|---|---|---|---|
| Flux Solution Method | CPU | Real | CPU | Real |
| Weighted Difference | 0.19 | 0.21 | 2.2 | 6.0 |
| Nodal | 0.44 | 0.57 | 7.2 | 13.7 |
| Characteristic | 1.79 | 1.82 | 6.2 | 12.8 |

It should be noted that these data represent typical user experience, and not necessarily the maximum capability of a given machine. In particular, the poor performance of the characteristic method on the Cray is due to the lack of funds with which to vectorize it. The IBM RS/6000 Model 320 workstation is now obsolete, and new workstations are many times faster. Additional performance data on newer machines are distributed with the source material.

## 1.7 UNUSUAL FEATURES OF THE PROGRAM

TORT is a three-dimensional successor to the two-dimensional DORT and DOT codes. Its basic methods have been field-tested in vigorous application dating back to about 1965. In the field of deep penetration problems, this family of codes is considered robust and dependable by many users worldwide. The new nodal and characteristic procedures have been especially adapted to minimize flux distortions that may occur in problems having irregular mesh shapes or wide variation in density.

Anisotropic scattering is represented by Legendre expansions of arbitrary order, and methods are available to repair erroneous scattering estimates caused by finite truncation of the expansion. Direction sets can be biased, concentrating work into directions of particular interest. Different direction sets can be used in various energy groups, allowing a finer directional definition where needed. Similarly, the order of Legendre expansion can vary with energy group.

A discontinuous space mesh capability allows the specification of mesh boundaries on adjacent rows of the mesh that do not match, and of planes whose cell boundaries do not match. This can be used, again, to concentrate the work in an area of specific need.

The coding is arranged such that special software such as high-performance disk file routines or run-time memory allocation can be implemented readily on computers that support them. Internal language flags allow the procedures in compute-intensive loops to be tailored to specific needs. In some cases, alternate subroutines are employed to improve efficiency. These features are selected and controlled by the install-time procedures. With these provisions, a single "version" serves all machine types.

## 1.8 RELATED AND AUXILIARY PROGRAMS

GIP Prepares cross section input

DORT Calculates two-dimensional boundary fluences for TORSED

TORSED Couples DORT fluences to external TORT boundaries

VISA Reformats and sorts DORT data for TORSED

TORSET Couples TORT fluences to a second TORT problem

DRV Coordinates various worker codes in a single job stream

## 1.9 STATUS

TORT is used on a routine basis by numerous installations worldwide. Updated versions are released from Oak Ridge roughly once a year.

## 1.10 REFERENCES

W. A. Rhoades and R. L. Childs, "The DORT Two-Dimensional Discrete Ordinates Transport Code," *Nucl. Sci. & Engr.* 99, 1, pp. 88-89 (May 1988).

W. A. Rhoades and R. L. Childs, "TORT: A Three-Dimensional Discrete Ordinates Neutron/Photon Transport Code," *Nucl. Sci. & Engr.* 107, 4, pp. 397-398 (April 1991).

W. A. Rhoades and R. L. Childs, "An Updated Version of the DOT 4 One- and Two-Dimensional Neutron/Photon Transport Code," ORNL-5851 (July 1982).

## 1.11 HARDWARE REQUIREMENTS

The code, itself, requires about 0.1 Megawords of memory. No overlay is used. An additional 0.1 Megawords of data storage is sufficient for modest problems. Problems such as the 104,247 mesh cells case cited above require roughly 1 Megaword of memory and 30 Megawords of disk space.

## 1.12 PROGRAMMING LANGUAGES

TORT is operable with 100% Fortran 77. On Cray, two optional assembler routines (3%) provide added speed. On UNIX workstations, optional C routines (roughly 0.3%) provide time, date, and location information, as well as run-time memory allocation.

## 1.13 OPERATING SYSTEM

The standard UNIX operating systems and compilers are used on the various machines.

## 1.14 OTHER PROGRAMMING OR OPERATING INFORMATION OR RESTRICTIONS

C Shell procedures are provided for unloading and installing the codes under UNICOS or AIX. Assembler and C routines are optional, as noted above. No proprietary or sensitive software is used. No restriction as to use, modification, or exchange of the material is made, except that programmers making modifications are requested to note that fact prominently in the version identification that is printed at the top of each job.

## 1.15 NAME AND ESTABLISHMENT OF AUTHOR OR CONTRIBUTOR

TORT was originated by W. A. Rhoades and R. L. Childs of Oak Ridge National Laboratory. Additions to and maintenance of TORT since the beginning of 1995 have been provided by Y. Y. Azmy, T. J. Burns, and D. B. Simpson. Other major contributors to Oak Ridge transport methodology and codes include W. W. Engle, Jr., F. R. Mynatt, E. T. Tomlinson, and D. R. Vondy.

Distribution is provided by:

Radiation Shielding Information Center (RSIC)
Oak Ridge National Laboratory
P.O. Box 2008, Building 6025
Oak Ridge, TN 37831-6362
(615) 574-6176 (Voice)
(615) 574-6182 (Fax)
http://epicws.epm.ornl.gov

RSIC charges a fee for handling, and they can often provide help with installation difficulties. Since no general support funds are available, telephone consultation with the originators is not provided. E-mail or letter reports of errors or difficulties will be considered as time permits.

## 1.16 MATERIAL AVAILABLE

The distribution includes files containing source coding, installation instructions, comments, document updates, a tutorial guide to the demonstration problems, problem input, and problem output, in addition to printed documentation.

## 1.17 KEYWORDS

TRANSPORT, NEUTRONICS, SHIELDING, DISCRETE ORDINATES, SN

## 1.18 SPONSOR

The original development of TORT was sponsored by the Defense Nuclear Agency, U. S. Department of Defense. Since that time, other sponsors have contributed to the installation of special features to fit their needs.

# REGION 2. INTRODUCTION

## 2.1 BACKGROUND

Active interest in a three-dimensional (3-D) deterministic transport code at Oak Ridge dates as far back as 1971, but the computers, mathematical methods, and programming techniques of that day were not ready to provide a practical working tool at that time. A 1977 study group considered several alternative development paths, but it concluded that straightforward extension of the traditional methods used in DOT 4[1] would be the most attractive near-term approach. Preparations began in 1981 with studies of vector coding techniques and nodal solution methods of particular value to 3-D applications. Continued mathematical and program structure developments in 2-D brought the goal closer.

When direct work on TORT began in 1983, the Los Alamos THREETRAN[2] code had provided an initial proof-of-principle for 3-D discrete ordinates, and Japan's ENSEMBLE[3] code had shown success in radiation transport areas. The initial version of TORT was developed exclusively to calculate the penetration of radiation into large concrete buildings for the Defense Nuclear Agency (DNA). It was always intended, of course, that the code should be adaptable to other applications in the future. A review of recent conference papers will indicate that this has happened.

## 2.2 FEATURES AND PROBLEM CAPABILITY

TORT can accept either RΘZ or XYZ geometry, as well as several 2-D geometries. A new means of entering material zone descriptions helps with the task of specifying very large problems. A coarse mesh can be used to reduce the time and memory required for flux acceleration. Material zones and coarse meshes are specified without reference to the fine solution mesh, so that the fine mesh can be changed without disrupting these features.

Cross section input files are supplied in formats familiar to ANISN[4] and DOT users. Awkwardness in handling $P_\ell$ cross section sets in the previous codes has been eliminated. The order$_\ell$ of P expansion can be varied without changing the cross-section information.

Directional quadrature requirements are similar to DOT requirements, except that twice as many directions are needed because of the third dimension. Quadratures can be biased in order to emphasize directions of particular interest.

Traditional problem types are available: fixed-source, subcritical multiplication searches, direct $k_{eff}$ searches, and indirect criticality searches. Fixed sources can occur at boundaries or internal to mesh cells. The searches employ an acceleration technique borrowed from VENTURE.[5] Searches can start with either a distributed source guess or a flux guess from a previous problem. Convergence importance can be specified by zone, and convergence can be monitored as the problem progresses.

Key flux locations can be selected for special monitoring during iteration. Numerous controls over the iteration and acceleration processes allow the experienced user to solve very difficult problems.

The flux sweep can be conducted with either weighted-difference, nodal, or characteristic formulations. Safeguards are provided to limit the effects of overextrapolation and flux distortion in difficult problems. A means of mitigating the negative sources produced by the finite cross-section expansion is provided. Flux acceleration is accomplished by the partial-current rebalance (PCR) method. Although this is less sophisticated than some later procedures, it is robust and compatible with TORT's various numerical procedures.

Traditional flux printout maps can be obtained, and flux can be printed at key positions. Group-integral responses can be obtained at key positions, by mesh cell, or by zone integral. A flux output file can be obtained for restarting and for use in preparing data summaries. The response information can be obtained as a file for use in peripheral codes.

## 2.3 PROGRAM STRUCTURE

TORT operates on many types of UNIX computers, from large parallel-vector Cray mainframes to sequential-scalar workstations such as the IBM RISC or DEC Alpha machines. Run-time memory allocation is used where available. Direct-access files are used extensively for scratch storage, and some of these can be saved for later use.

TORT is designed to operate in stand-alone mode, given enough input files, or as a subordinate module under the Discrete Ordinates System (DOS) driver.[6] (Unfortunately, the acronym "DOS" was subsequently used in another context by another supplier of software.) The code is structured in four major sections coupled only by data in the common blocks and scratch files. These sections could become individual overlays if necessary. The main body of the program has been kept as free of machine-dependent practices as practical. Exceptions to this policy are localized and identified by language flags. These flags are used to select the appropriate procedures when the code is installed. Other machine-dependent procedures and coupling to system library routines are localized in a drop-in adaptation package.

Only one source file is kept for the code, so that all users have equal access to corrections and improvements as they are made, regardless of the computing system they are using. Each printout is labelled with a version identifier unique to that specific release. The developers never authorize a change without a new identifier, assuring traceability of results.

## 2.4 THIS DOCUMENT

The remainder of this document is divided into theoretical, programmer's, user's, and environmental information regions. The theoretical information presents concepts and techniques used by TORT, relying upon numerous references for completeness. The programmer's region concentrates on

information helpful in understanding the code structure. The user's information provides specific instructions for preparing problem input data. It is intended to be thorough enough to guide a novice user with some experience in transport applications. The environmental region describes the adaptation to specific systems.

## 2.5 ADDITIONAL INFORMATION AVAILABLE

In addition to the information presented here, a bibliography of related Oak Ridge publications and papers is included at the end of this report. Last-minute information is distributed in files supplied with the source program. In this form, it can be kept timely and informal, and then changed or replaced when the situation demands. This information typically includes applications notes giving specific advice about installing the code on a given system, a list of applicable documentation, updates and corrections to the documentation, a discussion of additional code features or limitations, and supplemental information useful in applying the code.

## 2.6 FUTURE DIRECTIONS

While it is not unusual for a code to be funded as an objective in itself, e.g. DOT 4, TORT has arisen from the demands of specific applications projects. Consequently, there remain many areas where additional effort would produce important improvements in performance and generality. Many of the requirements will emerge as the code is applied to new areas -- the DOT series of codes emerged from about 15 years of interaction between users and program developers.

Assistance could be provided to users in selecting an appropriate space mesh and in displaying the adequacy of a chosen mesh. Improvements in controlling the effects of finite Legendre expansions would be welcome. The acceleration of flux and source iterations would benefit from improvement. A coordinated package for graphical flux and response display would be valuable.

Cray users would benefit from a vectorized characteristic flux routine. Parallel operation on a Cray system was programmed in 1995. Operation on other types of parallel computers, especially those that do not require radical revision of the algorithms, could be fruitful.

## 2.7 LIMITATIONS AND CORRECTIONS

Certain items described in the manual are not available and ready for use in Version 3.1, the reference version at the time of this writing. These include:

- linear-zero flux calculations,
- cylindrical, white, and albedo boundary conditions,
- upscatter rebalance,
- indirect searches and super-meshes, and
- boundary sources on internal i- and j-boundaries.

# REGION 3. THEORETICAL BASIS

## 3.1 THE BOLTZMANN TRANSPORT EQUATION

Deterministic radiation transport codes such as TORT depend upon the Boltzmann transport equation as the basic mathematical model. The equation is, in principle, just a statement of local balance between currents, disappearance due to collision, and sources produced from material interaction or from an external process. It is attributed to L. Boltzmann, who used it in connection with the kinetic theory of gases.[7] Weinberg and Wigner[8] relate the basic equation to a 1910 publication by Boltzmann,[9] and discuss its application to the physical processes involved in the transport of neutrons. Given proper cross section data, photon transport can be described as well. With time-dependence suppressed, the form pertinent to neutron/photon transport in a fissile system can be expressed as:

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

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, $\psi$, is defined such that the number of particles within a small spherical volume $d\underline{r}$ about $\underline{r}$ at time $t$, with energies falling within $dE$ and directions of motion within $d\underline{\Omega}$, will be $\frac{1}{v}\psi\, d\underline{\Omega}\, dE\, d\underline{r}$. Also, $\sigma^T$ and $\sigma^F$ are macroscopic cross sections for total interaction and fission, respectively, and $\sigma^S$ is the cross section for scattering from energy $E'$ and direction $\underline{\Omega}'$ to energy $E$ and direction $\underline{\Omega}$. The functions $\nu$ and $\chi$ 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. As a practical matter, scattering is generally independent of initial direction, so that it can be expressed in terms of the single scattering angle,

$$\mu_0 \equiv \underline{\Omega}' \cdot \underline{\Omega} \quad , \tag{3.2}$$

rather than as a function of two direction vectors. Although some applications exist for the

dependence of $\sigma^T$ on $\underline{\Omega}$, they are unusual, and this dependence is not allowed in the present work. A somewhat weaker justification exists for the removal of the dependence of $\chi$ on position in space. In fact, if several fissile species exist in the system, and if the species do not always fission in the same proportion, $\chi$ will depend upon position. Nonetheless, the present work does not allow such dependence.

## 3.2 NUMERICAL SOLUTION OF THE TRANSPORT EQUATION

### 3.2.1 The Discrete Ordinates Formulation

Without the integral over direction space on the right-hand side, the solution of 3.1 would be fairly straightforward. Integrating over energy produces a set of multigroup equations with group-integral fluxes interacting according to mean cross section values. The spatial dependence can be represented with a set of ordinary algebraic equations relating cell-average flux values and boundary currents. It is the directional coupling through the source integral that brings most of the solution difficulty.

In the method of discrete ordinates, the equation is evaluated over a specific set of discrete directions, and the integral is approximated by a weighted sum over the directional results. Once again, a set of algebraic equations is obtained, and these can be solved by a combination of recursion and iteration procedures. A 1957 book by Davison[10] gave a review of the method and an appendix citing important developmental work by Carlson at Los Alamos. Davison cites a 1953 book by Chandrasekhar[11] as the basis of the method as he presents it, and a 1943 article by Wick[12] as the origin of the basic approach.

The discrete ordinates method as we know it is based upon the "discrete $S_n$ method" reported by Carlson and Bell in 1958.[13] This was the second of two $S_n$ methods described in the paper, but it had so many advantages over the first, and was so well adapted to computing machinery, that the first method was soon abandoned. In addition to conserving particles without special fixups, the discrete method was readily extendable to multiple space dimensions and anisotropic scattering treatments. Its accuracy could, in general, be improved by adding more directions, and the solution work increased no faster than linearly with the number of discrete directions.

A 1965 report by Carlson and Lathrop[14] gave a review of the method with emphasis on physical principles and evaluation techniques as well as an annotated bibliography of earlier work in the field. Mynatt *et al.*[15] provided a formal mathematical development of RZ geometry in 1969, while Lathrop and Brinkley[16] gave both RZ and R$\Theta$ formulations in 1973. In this report, we follow the notation of Lathrop and Brinkley, although Mynatt's results are equivalent. In three dimensions, either R$\Theta$Z or XYZ geometry, the form we use is:

$$
\begin{aligned}
& W_m \mu_m \left[ A_{i+1/2,j,k} N_{i+1/2,j,k,m,g} - A_{i-1/2,j,k} N_{i-1/2,j,k,m,g} \right] \\
& \quad + W_m \xi_m \left[ B_{i,j+1/2,k} N_{i,j+1/2,k,m,g} - B_{i,j-1/2,k} N_{i,j-1/2,k,m,g} \right] \\
& \quad + W_m \eta_m \left[ C_{i,j,k+1/2} N_{i,j,k+1/2,m,g} - C_{i,j,k-1/2} N_{i,j,k-1/2,m,g} \right] \\
& \quad + \left[ A_{i+1/2,j,k} - A_{i-1/2,j,k} \right] \left[ \alpha_{m+1/2} N_{i,j,k,m+1/2,g} - \alpha_{m-1/2} N_{i,j,k,m-1/2,g} \right] \\
& \quad + \sigma^T_{i,j,k,g} W_m V_{i,j,k} N_{i,j,k,m,g} = W_m V_{i,j,k} S_{i,j,k,m,g} \quad .
\end{aligned}
\tag{3.3}
$$

In this notation, subscripts *i, j,* and *k* represent mesh intervals in the three space dimensions. The subscript *m* refers to one of an 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 terms with subscripts such as *m+1/2* 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 Eq. (3.3) might later be written:

$$
W\mu \left[ A_{i+1/2} N_{i+1/2} - A_{i-1/2} N_{i-1/2} \right] \quad .
$$

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

In certain discussions, *I, J,* and *K* are understood to be the upper limits of *i, j,* and *k*. 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, j=J+1/2, k=1/2,* and *k=K+1/2* are spoken of as "left," "right," "inside," "outside," "bottom," and "top," boundaries, regardless of geometry.

The ordered set of directions of particle travel are characterized by their direction cosines ($\mu_m$, $\xi_m$, $\eta_m$). In all geometries, $\mu$ is the cosine of the angle that the direction of travel makes with first-dimension axis, i.e., X or R, while $\eta$ is the cosine of the angle with the Z axis, and $\xi$ is the cosine with the remaining direction vector, either the Y axis or the azimuthal axis in the case of R$\Theta$Z geometry.

The coefficients *A, B,* and *C* are cell face areas perpendicular to the axes from which $\mu$, $\xi$, and $\eta$ are measured. The $\alpha$ coefficients will be defined later.

In the geometries of interest, *B* and *C* have no variation across a cell, leading to some simplification. We have reversed the roles of $\xi$ and $\eta$ from Carlson's original usage so that $\eta$ could continue being associated with the Z direction, as it was in RZ geometry. Our direction-cosine ordering and naming of boundaries are consistent with the ENSEMBLE code.[3]

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, $\psi$, 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 $\sigma^T$ 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).*

It can be observed that, except for the terms with $\alpha$, Eq. (3.3) 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 $\alpha$ terms are defined appropriately.

The terms with $\alpha$ are effective only in curved geometry, R$\Theta$Z. Their physical meaning can be made apparent by considering a particle moving inward along a path that will take it closer to the axis of a cylinder. While $\eta$ is a constant for such a path, $\mu$ 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 $\mu$ for these paths.

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

The recursion relationship for the $\alpha$'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 $\sigma^T$, *A*, *B*, *C*, or *V*, then it must follow that, for a given $\eta$ level,

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

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

$$\sum_m W_m \mu_m = 0 \quad , \tag{3.5}$$

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 $\eta$ level. This ensures that particle conservation is rigorously met. Symmetrical quadrature sets inherently obey this condition, and it is important that unsymmetrical sets obey this condition as well.

It is advantageous to substitute a new variable, $\beta$, for $\alpha$:

$$\beta_m \equiv \frac{1}{W_m}\left[\alpha_{m+1/2} + \alpha_{m-1/2}\right] \quad . \tag{3.6}$$

The recursion relationship for the $\beta_m$ can be obtained by combining its definition and the recursion relationship for $\alpha$, each evaluated at both $m$ and $m+1$:

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

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

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

and

$$\alpha_{m+3/2} = \alpha_{m+1/2} - W_{m+1} \mu_{m+1} \tag{3.10}$$

Therefore,

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

Recalling that $\alpha_{m-1/2}=0$ for the first direction of each $\eta$ level, then it follows that $\beta_m=-\mu_m$ for such levels, at least if $W_m \neq 0$. This relationship is also assumed 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 $i$ values toward small, and likewise with the other dimensions. Similarly, positive $\mu$ values require

evaluation from small $i$ 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:

$$c \equiv (1/2)sg(\mu_m) \; ; \quad d \equiv (1/2)sg(\xi_m) \; ; \quad e \equiv (1/2)sg(\eta_m) \; . \tag{3.12}$$

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

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

where

$$\Delta A \equiv A_{i+1/2} - A_{i-1/2} \; , \tag{3.14}$$

where unessential subscripts have been dropped, and where it is recognized that $B$ and $C$ are independent of $j$ and $k$, respectively.

### 3.2.2 Geometric Coefficients

The geometric unit cells used in the code are illustrated in Fig. 3.1 (pg. 3-53). The faces corresponding to the geometric parameters $A$, $B$, and $C$ are shown. Table 3.1 gives the values of the coefficients in terms of cell dimensions. It should be noted that *the theta dimension is measured in units of rotations, not radians;* thus, theta varies from 0 to 1 during a full rotation.

**Table 3.1. Geometric Coefficients**

| | XYZ | RΘZ |
|---|---|---|
| A | ΔY·ΔZ | 2πR·ΔΘ·ΔZ |
| B | ΔX·ΔZ | ΔR·ΔZ |
| C | ΔX·ΔY | $2\pi\overline{R}$·ΔR·ΔΘ |
| V | ΔX·ΔY·ΔZ | $2\pi\overline{R}$·ΔR·ΔΘ·ΔZ |

NOTES:

1. Θ is measured in rotations.
2. $\overline{R}$ is the value of R midway between faces.

### 3.2.3 Basic Flux Evaluation Strategy: The "Linear" or "Diamond" Difference Model

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 $i$, $j$, and $k$. Where $I$, $J$, and $K$ represent the largest values of $i$, $j$, and $k$, then the boundary values $N_{I+1/2}$, $N_{J+1/2}$, and $N_{K+1/2}$ 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 $m=1/2$, let us observe that, even with $N_{I-c}$, $N_{J-d}$, and $N_{K-e}$, known from boundary considerations and $N_{m-1/2}$ to be defined by a yet-unspecified process, Eq. (3.13) would still involve five unknowns. The four advance boundary values, $N_{I+c}$, $N_{J+d}$, and $N_{K+e}$ $N_{m+1/2}$ have been treated as completely independent, and are, therefore, not completely defined by Eq. (3.13). In fact, all of these represent values of a single continuous function, $\psi$, at adjacent locations, and they cannot vary more freely than $\psi$ varies. If we assume that $\psi$ can be approximated adequately by straight-line segments between adjacent boundaries, and if the average value $N$ is simply the linear average of the boundary values, then the following "linear" or "diamond" difference model results:

$$N = (1/2)\left[N_{i+c} + N_{i-c}\right] , \tag{3.15a}$$

$$N = (1/2)\left[N_{j+d} + N_{j-d}\right] , \tag{3.15b}$$

$$N = (1/2)\left[N_{k+e} + N_{k-e}\right] , \tag{3.15c}$$

and

$$N = (1/2)\left[N_{m+1/2} + N_{m-1/2}\right] . \tag{3.15d}$$

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

$$N_{3/2} = N_1 . \tag{3.16}$$

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

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

Where an initiating direction occurs for $m>1$, and where $m'$ denotes the predecessor to $m$, then Eq. (3.15) defines values for $m'+1/2$ and $m-1/2$ that do not agree. In fact, such directions are on different η levels, and they do not, in general, describe adjacent directions. Thus, there is neither a physical nor a 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.[14] On the other hand, the TWOTRAN II code[16] uses directions having non-zero but small, finite weights. Tomlinson *et al.*[17] showed that, other factors being equal, this choice does not significantly affect the results of problems they studied.

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:

$$N_{i+c} = 2N - N_{i-c} \quad , \tag{3.18a}$$

$$N_{j+d} = 2N - N_{j-d} \tag{3.18b}$$

$$N_{k+e} = 2N - N_{k-e} \quad , \tag{3.18c}$$

and

$$N_{m+1/2} = N \quad . \tag{3.18d}$$

Defining, as a matter of convenience:

$$\overline{A} \equiv (1/2)\left[A_{i+1/2} + A_{i-1/2}\right] \quad , \tag{3.19}$$

observing that:

$$2|\mu| A_{i+c} + \Delta A\left[\beta - \mu\right] = 2|\mu|\overline{A} + \Delta A\beta \quad , \tag{3.20}$$

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

$$N = \frac{VS + 2|\mu|\,\bar{A}N_{i-c} + 2|\xi|\,BN_{j-d} + 2|\eta|\,CN_{k-e}}{V\sigma^T + 2|\mu|\,\bar{A} + 2|\xi|\,B + 2|\eta|\,C} \;;\; \beta = -\mu \quad . \tag{3.21}$$

This equation applies to all 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. (3.15d) to the preceding directions:

$$N_{m+1/2} = 2N - N_{m-1/2} \;;\; \beta \neq -\mu \quad . \tag{3.22}$$

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

$$N = \frac{VS + 2|\mu|\,\bar{A}N_{i-c} + 2|\xi|\,BN_{j-d} + 2|\eta|\,CN_{k-e} + \Delta A\beta N_{m-1/2}}{V\sigma^T + 2|\mu|\,\bar{A} + 2|\xi|\,B + 2|\eta|\,C + \Delta A\beta} \;;\; \beta \neq -\mu \quad . \tag{3.23}$$

At this point, an explicit sweep pattern has been found. Determining $N_{I+1/2}$, $N_{J+1/2}$, $N_{K+1/2}$, and $N_{m+1/2}$ as described, Eq. (3.21) gives $N_{I,J,K,l}$ explicitly. Equation (3.18a) 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 this done, values for $N_{m+1/2}$ for each initiating direction can be found using Eq. (3.17). Similarly, additional directions with $\mu<0$ in the given $\eta$ level are evaluated from Eq. (3.23) and extrapolated with Eq. (3.22). This process continues for all $m$'s for which $\eta<0$, $\xi<0$, and $\mu<0$. Having these values, the values for directions having $\eta<0$, $\xi<0$, 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. (3.18b) determines $N_{J-1/2}$, and the process proceeds as before, inward to $j=1/2$. Boundary values at $j=1/2$ for directions having $\eta<0$ and $\xi>0$ are determined from physical boundary conditions, and the sweep resumes, this time outward from $j=1/2$ to $j=J+1/2$. The sweep in $K$ proceeds analogously; downward and then upward.

It is clear from physical considerations that, given positive sources, boundary values, and cross sections, we expect $N$ to be everywhere positive. Certainly Eqs. (3.21) and (3.23) will produce positive $N$'s under such conditions. Unfortunately the "extrapolation equations," (3.18), will not always do so. If:

$$N < (1/2)\max\left[N_{i-c}, N_{j-d}, N_{k-e}, N_{m-1/2}\right] \;, \tag{3.24}$$

then at least one value produced by the extrapolations will be negative. This may result in negative

values of $N$ in subsequent applications of Eq. (3.21) or Eq. (3.23). These false values result from 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. (3.21), the value of $N$ can be made arbitrarily close to 0 by a large value of $\sigma^T$, as is well known. Also, if $S$, $N_{j-d}$, and $N_{k-e}$ are 0, but either $B$ or $C$ is not 0, negative values of $N_{i+c}$ will result for all directions having:

$$|\mu|\,\bar{A} < |\xi|\,B + |\eta|\,C \tag{3.25}$$

*for any value of $\sigma^T$, even $\sigma^T=0$!* If false negatives are so easily generated, it is apparent that inaccurate positive numbers can also be generated, and these errors are not so easily detected. We call this difficulty "mesh distortion," since it results from interaction of the mesh shape with the numerical method, whether material is present or not. These problems render Eqs. (3.21) and (3.23) unsuitable for use as given. More complexity is the only remedy, as will be seen.

### 3.2.4 An Inherently Positive Method: The Step Model

The "step" model is much simpler. If the approach of Eq. (3.17) is arbitrarily extended to all extrapolations, the result is:

$$N_{i+c} = N \quad , \tag{3.26a}$$

$$N_{j+d} = N \quad , \tag{3.26b}$$

$$N_{k+e} = N \, , \tag{3.26c}$$

and

$$N_{m+1/2} = N \quad . \tag{3.26d}$$

Then, Eq. (3.13) can be solved for $N$ directly. With a bit of simplification, if $\beta \neq -\mu$:

$$N = \frac{VS + |\mu|\,A_{i-c}N_{i-c} + |\xi|\,BN_{j-d} + |\eta|\,CN_{k-e} + (1/2)\,\Delta A(\beta+\mu)N_{m-1/2}}{V\sigma^T + |\mu|\,A_{i-c} + |\xi|\,B + |\eta|\,C + (1/2)\,\Delta A(\beta+\mu)} \quad , \tag{3.27a}$$

and if $\beta = -\mu$:

$$N = \frac{VS + |\mu|\,A_{i-c}N_{i-c} + |\xi|\,BN_{j-d} + |\eta|\,CN_{k-e}}{V\sigma^T + |\mu|\,A_{i-c} + |\xi|\,B + |\eta|\,C} \quad . \tag{3.27b}$$

Given values of the incoming boundary fluxes, Eq. (3.27) provides the average flux, Eq. (3.26) gives the outgoing boundary fluxes, and the calculation proceeds. The step model can be seen to yield positive results, given positive sources and boundary fluxes. It will be seen later that it approaches the correct limit for thick intervals. Unfortunately, its accuracy is deficient for small and intermediate cells.

### 3.2.5 Fixup Methods: Linear Zero and Weighted Difference

The strengths and weaknesses of the linear and step models can be seen by examining the limiting case of 1-D, source-free plane geometry, in which $\Delta A=0$ and $S$=0. Letting:

$$V = \bar{A}\,\Delta X \quad ,$$

Eqs. (3.18a) and (3.23) can be solved for emerging flux, yielding:

$$r^L \equiv N_{i+c}/N_{i-c} = \frac{1-(1/2)s}{1+(1/2)s} \quad , \tag{3.28}$$

where:

$$s \equiv \frac{\sigma^T \Delta X}{|\mu|} \quad .$$

The exact solution in such a case is shown in Appendix E to be:

$$r^x \equiv e^{-s} = 1 - s + (1/2)s^2 - (1/6)s^3 + \ldots \quad . \tag{3.29}$$

Expanding $r^L$ in a Maclaurin series yields:

$$r^L = 1 - s + (1/2)s^2 - (1/4)s^3 + \ldots \quad , \tag{3.30}$$

which can be seen to be correct through the term in $s^2$, i.e., "second-order correct," in this idealized case. Appendix E also examines the case of a source varying in space.

The step model for the case examined above yields:

$$r^s = \frac{1}{1+s} = 1 - s + s^2 + \ldots \quad , \tag{3.31}$$

which is accurate only to terms in $s$, i.e., "first-order correct." Table 3.2 shows selected values of these functions. As one would expect, Eq. (3.28) is very accurate with $s$ small, with just 1% error at $s=1/2$. Unfortunately, high-order accuracy with $s$ small is accompanied by high-order error with $s$ large. The errors are both about 12% of the original flux at $s=1$, but the linear method dips to comparatively large negative values after $s=2$, while the step method continues smoothly, although slowly, to 0. The large, non-physical negatives can cause wrong overall results and can thwart convergence.

An average of the two methods produces accurate results between $s=0$ and $s=2$. Unfortunately, large negative values are produced at large $s$. Such difficulties plague any attempt at a fixed formula blending the two methods. From this, however, one can see the requirements of a good intermediate method. First, it should possess the accuracy of the linear method for small intervals, so that the dominant portions of the calculation can be accurate. Second, it should go to the correct limits for thick intervals. Finally, it should yield acceptably small, non-negative results for cells where mesh distortion and shadowing tend to produce errors. With such a compromise, fluxes crossing a wide interval at an oblique angle, for example, would typically be overwhelmed by correctly calculated fluxes arriving from an easier path, and they would do no harm.

**Table 3.2. Comparison of Linear and Step Results**

| $s$ | Linear $r^L$ | Exact $r^x$ | Step $r^s$ | Average $(1/2)(r^L+r^s)$ |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1/2 | 0.60 | 0.61 | 0.67 | 0.64 |
| 1 | 0.25 | 0.37 | 0.50 | 0.38 |
| 1 1/2 | 0.14 | 0.22 | 0.40 | 0.27 |
| 2 | 0 | 0.14 | 0.33 | 0.16 |
| 3 | -0.20 | 0.050 | 0.25 | 0.02 |
| 4 | -0.33 | 0.018 | 0.20 | -0.06 |
| 8 | -0.60 | 0 | 0.11 | -0.24 |
| ∞ | -1 | 0 | 0 | -0.50 |

Early codes such as DTK[18] attempted to achieve such behavior by checking the calculation at each interval and substituting a step calculation if a negative extrapolated flux was detected. This procedure worked fairly well in one dimension. In 2-D, however, problems would frequently experience "switching oscillation," in which cells not too far from each other would "resonate," alternately switching between models, and thus thwarting convergence. Further, the calculation of flux across the width of a long, narrow interval would be inaccurate due to the use of the step model to prevent negative fluxes along the length.

Lathrop and Brinkley[16] had a better idea for TWOTRAN II. They simply set the negative flux to zero and re-solved for the other fluxes with that assumption. This procedure, sometimes called "zero fixup" or "linear-zero," brought acceptable stability and convergence to the solution of difficult problems, and it was included in later versions of DOT.[1] An important advantage over the linear-step method was that the result was continuous at $s=2$, whereas the linear step suffered a discontinuity there. A disadvantage was that zero was always an underestimate of the true result – undesirable in many applications. While linear-zero was a simple procedure in 1-D, it required a great deal of additional coding in 2-D. In fact, bugs were found in the procedure years after its introduction.[19] Extending the 2-D method to 3-D is even more complicated, and the coding does not vectorize readily.

The weighted-difference model[20] uses a different approach. Rather than looking for negative results experimentally, the formulation seeks to blend linear and step models in such proportions as to provide linear-model accuracy with $s$ small and to make a smooth transition to zero where over-extrapolation or mesh distortion is possible. Further, when the realities of sources and incoming boundary fluxes are added, one can adjust certain parameters in an attempt to improve accuracy. Although the initial acceptance of weighted difference was slow, the method has dominated deep-penetration calculations since the mid-1970's, and has yet to be displaced in curved geometry. It is easily adapted to 3-D and largely vectorizable. Although it requires more computational work per iteration than linear-zero, it is often more economical due to faster convergence.[21]

To derive the weighted-difference models, define, where $\beta \neq \mu$:

$$\gamma^i \equiv \frac{N - N_{i-c}}{N_{i+c} - N_{i-c}} \quad , \tag{3.32a}$$

$$\gamma^j \equiv \frac{N - N_{j-d}}{N_{j+d} - N_{j-d}} \quad , \tag{3.32b}$$

$$\gamma^k \equiv \frac{N - N_{k-e}}{N_{k+e} - N_{k-e}} \quad , \tag{3.32c}$$

and

$$\gamma^m \equiv \frac{N - N_{m-1/2}}{N_{m+1/2} - N_{m-1/2}} \quad , \tag{3.32d}$$

from which it follows that:

$$N_{i+c} = \frac{1}{\gamma^i} N - \left[\frac{1-\gamma^i}{\gamma^i}\right] N_{i-c} \quad , \tag{3.33a}$$

$$N_{j+d} = \frac{1}{\gamma^j} N - \left[\frac{1-\gamma^j}{\gamma^j}\right] N_{j-d} \quad , \tag{3.33b}$$

$$N_{k+e} = \frac{1}{\gamma^k} N - \left[\frac{1-\gamma^k}{\gamma^k}\right] N_{k-e} \quad , \tag{3.33c}$$

and

$$N_{m+1/2} = \frac{1}{\gamma^m} N - \left[\frac{1-\gamma^m}{\gamma^m}\right] N_{m-1/2} \quad . \tag{3.33d}$$

With these substitutions into Eq. (3.13):

$$N = \frac{VS + F^i N_{i-c} + F^j N_{j-d} + F^k N_{k-e} + F^m N_{m-1/2}}{V\sigma^T + F^i + F^j + F^k + F^m} \quad , \tag{3.34}$$

where

$$F^i = |\mu| \left[(1/\gamma^i - 1) A_{i+c} + A_{i-c}\right] \quad , \tag{3.35a}$$

$$F^j = |\xi| \left(1/\gamma^j\right) B \quad , \tag{3.35b}$$

and

$$F^k = |\eta| \left(1/\gamma^k\right) C \quad , \tag{3.35c}$$

$$F^m = (1/2)\,\Delta A\left[\left(1/\gamma^m - 1\right)(\beta - \mu) + (\beta + \mu)\right] \quad . \tag{3.35d}$$

The values of $\gamma^i$, $\gamma^j$, $\gamma^k$, and $\gamma^m$ are determined by substituting Eq. (3.33) into Eq. (3.34), and then, by examination, choosing each weighting coefficient to ensure non-negativity of the boundary flux for any value of the other coefficients between 1/2 and 1. (The reason for the 1/2 and 1 limits will be seen soon.) Sufficient conditions are:

$$1-\gamma^i = \frac{SV\theta_s + \left[|\xi|\, BN_{j-d} + |\eta|\, CN_{k-e}\right]\theta_n + \left[(1/2)\Delta A(\beta+\mu)N_{m-1/2}\right]\theta_m + |\mu|\, A_{i-c}N_{i-c}}{2\left[(1/2)V\sigma^T + |\xi|\, B + |\eta|\, C + (1/2)\Delta A(\beta-\mu)\right]N_{i-c}} \quad , \tag{3.36a}$$

$$1-\gamma^j = \frac{SV\theta_s + \left[|\mu|\, A_{i-c}N_{i-c} + |\eta|\, CN_{k-e}\right]\theta_n + \left[(1/2)\Delta A(\beta+\mu)N_{m-1/2}\right]\theta_m + |\xi|\, BN_{j-d}}{2\left[(1/2)V\sigma^T + |\mu|\, A_{i+c} + |\eta|\, C + (1/2)\Delta A(\beta-\mu)\right]N_{j-d}} \quad , \tag{3.36b}$$

$$1-\gamma^k = \frac{SV\theta_s + \left[|\mu|\, A_{i-c}N_{i-c} + |\xi|\, BN_{j-d}\right]\theta_n + \left[(1/2)\Delta A(\beta+\mu)N_{m-1/2}\right]\theta_m + |\eta|\, CN_{k-e}}{2\left[(1/2)V\sigma^T + |\mu|\, A_{i+c} + |\xi|\, B + (1/2)\Delta A(\beta-\mu)\right]N_{k-e}} \quad , \tag{3.36c}$$

and

$$1-\gamma^m = \frac{SV\theta_s + \left[|\mu|\, A_{i-c}N_{i-c} + |\xi|\, BN_{j-d} + |\eta|\, CN_{k-e}\right]\theta_n + (1/2)\Delta A(\beta+\mu)N_{m-1/2}}{2\left[(1/2)V\sigma^T + |\mu|\, A_{i+c} + |\xi|\, B + |\eta|\, C\right]N_{m-1/2}} \quad , \tag{3.36d}$$

where $\theta_m$, $\theta_n$, and $\theta_s$ are arbitrary parameters between 0 and 1. It can be shown that the linear model corresponds to:

$$\gamma^i = \gamma^j = \gamma^k = \gamma^m = 1/2 \quad , \tag{3.37}$$

while the step model corresponds to:

$$\gamma^i = \gamma^j = \gamma^k = \gamma^m = 1 \quad . \tag{3.38}$$

When $\beta$=-$\mu$, the equation for $N$ is simplified as usual. Substituting Eqs. (3.33a), (3.33b), and (3.33c), together with Eq. (3.17) into Eq. (3.13):

$$N = \frac{VS + F^i N_{i-c} + F^j N_{j-d} + F^k N_{k-e}}{V\sigma^T + F^i + F^j + F^k} \quad . \tag{3.39}$$

The weighting coefficients are derived in the previous manner, giving:

$$1 - \gamma^i = \frac{SV\theta_s + \left[\, |\xi|\, BN_{j-d} + |\eta|\, CN_{k-e} \,\right] \theta_n + |\mu|\, A_{i-c} N_{i-c}}{2\left[ (1/2)\, V\sigma^T + |\xi|\, B + |\eta|\, C - (1/2)\, \Delta A \mu \,\right] N_{i-c}} \quad , \tag{3.40a}$$

$$1 - \gamma^j = \frac{SV\theta_s + \left[\, |\mu|\, A_{i-c} N_{i-c} + |\eta|\, CN_{k-e} \,\right] \theta_n + |\xi|\, BN_{j-d}}{2\left[ (1/2)\, V\sigma^T + |\mu|\, A_{i+c} + |\xi|\, B - (1/2)\, \Delta A \mu \,\right] N_{j-d}} \quad , \tag{3.40b}$$

and

$$1 - \gamma^k = \frac{SV\theta_s + \left[\, |\mu|\, A_{i-c} N_{i-c} + |\xi|\, BN_{j-d} \,\right] \theta_n + |\eta|\, CN_{k-e}}{2\left[ (1/2)\, V\sigma^T + |\mu|\, A_{i+c} + |\xi|\, B - (1/2)\, \Delta A \mu \,\right] N_{k-e}} \quad . \tag{3.40c}$$

The $\gamma^m$ coefficient is not defined in this case, and is not used.

With non-negative sources and incoming fluxes at the lateral boundaries, e.g., $N_{j-d}$, $N_{k-e}$, and $N_{m-1/2}$ when $\gamma^i$ is being calculated, it can be shown that the $\gamma$'s are never greater than 1. With lateral fluxes and sources zero, the positive step result will be calculated for large $V\sigma^T$. By limiting $\gamma$ to no less than 1/2, the results will be identical to the linear model for small $V\sigma^T$. In this case, the value of the $\theta$'s is irrelevant. It is in the treatment of sources and incoming fluxes at the lateral boundaries that the choice of $\theta$ becomes important.

The $\theta$'s are arbitrary, although it is intended that they lie between 0 and 1. The formulation discussed in the 1969 version of TWOTRAN (X,Y)[22] is equivalent to:

$$\theta_s = \theta_m = \theta_n = 0 \quad . \tag{3.41}$$

This method never generates negative results, but it will use the less-accurate step model for intervals large in one or more directions, even though flow across the cell faces and internal sources would allow use of the linear model.

A 1969 Lathrop paper[23] had:

$$\theta_s = \theta_m = \theta_n = 1 \quad . \tag{3.42}$$

This method maximizes use of the linear model, but it tends, in certain cases, to break down for exceedingly small fluxes, thwarting convergence. In the 1973 version of DOT III,[14] a compromise was used:

$$\theta_s = 1 \; ; \; \theta_m = \theta_n = 0 \quad . \tag{3.43}$$

This worked well in source-dominated regions, but gave inaccurate results for source-free intervals of long, narrow shape. The original weighted-difference model available in the earliest DOT IV,[17,25] now called "ordinary-" or "zero-weighted" difference, used:

$$\theta_s = \theta_m = 1 \; ; \; \theta_n = 0 \quad . \tag{3.44}$$

This mitigated certain distortions along the axes of cylinders, but the trouble with long, narrow intervals remained.

In a later development, the "theta-weighted"[17] model set:

$$\theta_s = \theta_m = \theta_n = \theta \quad , \tag{3.45}$$

where $\theta$ is empirically chosen such as to avoid degeneracy, and yet, to result in the usually-more-accurate linear model when the source or flow from adjacent boundaries allow it. Values of $\theta$ of 0.5 or greater, but not near enough to 1.0 to produce numerical degeneracy, give essentially equivalent results. It is not intended that this parameter should be "fine-tuned", and values of 0.5 or 0.9 are typical. If greater accuracy is required, another method should probably be used.

The TORT and DORT codes have a slightly modified version called "consistent weighted":

$$\theta_s = \theta_m = 1 \; ; \; \theta_n = \theta \quad . \tag{3.46}$$

When $\theta=0$, this is identical to the "zero-weighted" model. When $\theta>0$, results are virtually identical to the "theta-weighted" model of DOT IV. Only one set of coding is needed to produce both results.

This version of weighted difference has been in use since the early 1980's. It does, however, have a potentially important weakness. Let us examine Eq. (3.34) in the case of a thick interval in slab geometry, where terms involving $B$, $C$, and $\Delta A$ disappear. Then:

$$1 - \gamma^i = \frac{SV\theta_s}{V\sigma^T N_{i-c}} \quad .$$

This is indeterminate, of course, when $N_{i-c}=0$, but $\gamma^i$ is held at the lower limit of its range, 0.5, as $N_{i-c}$ becomes large, and it appears important to use that value at the limit, also.

As $S$ becomes very large and $N_{i-c}$ remains finite, $\gamma^i$ is also held at the limit 0.5. It is seen in Appendix E that this generally leads to poor results, and $\gamma^i$ should be held at its upper limit, 1.0, for thick intervals. This can be brought about by setting:

$$\theta_s = 0 \quad .$$

Experiments with this modified form show an important reduction in the number of cases that refuse to converge. The effect on the accuracy of the results is not known, however. Since the modified form would usually be correct for large intervals, the only concern would be for small intervals, where the new formulation might or might not be acceptable.

### 3.2.6 Semi-Analytical Methods: Nodal and Characteristic

All of the approximations discussed in the previous sections defined the average flux, $N$, as an arbitrary function of the boundary fluxes, and this brought certain persistent trouble areas. First, the linear or diamond-difference approximation that is sufficiently accurate for small mesh cells tends to behave badly when the flux changes rapidly across a mesh cell. Also, Appendix E shows that this approximation can give unacceptable results for thick mesh cells, even when boundary flow and source are nearly in equilibrium. Weighted difference can give acceptable boundary fluxes for both thick and thin intervals, but it is limited in its ability to represent flux within the cell, and thus, to calculate reaction rates correctly.

The semi-analytical methods to be presented here will use flat or linear representations of the spatial distribution of sources within the cell and fluxes at the boundary of the cell. (This was not a consideration in the previous work at all, since it dealt only with the overall cell balance.) Then, they will use analytical solutions of the discrete ordinates equation within the cell. The results will be more complex, but an improved ability to provide correct results for large mesh cells will be obtained. Given positive, constant sources and boundary fluxes, the results will be inherently positive. Additional constraints will force positive results with linear representation of the sources and boundary fluxes. The idea of using analytical formulations within a cell is not new. A 1969 paper by Lathrop dwelt at length on early applications of the concept to discrete ordinates.[23] The methods used in TORT are discussed in detail in a 1993 report.[26]

The general procedure used here is an application of "quasi-linearization," a process used as long ago as 1958 by Cohen,[27] and it is a natural extension of the Runge-Kutta method for solving initial value problems. In this procedure, terms in an equation like Eq. (3.13) that depend explicitly upon the independent variable, $N$ in this case, are separated from the "source" terms that drive it. The latter terms are approximated by low-order polynomials, and the resulting equation can be integrated

analytically. The source terms can have weak dependence on $N$, if their predominant behavior is independent. Cohen and Flatt reported considerable success in solving reactor kinetics problems with this method, using time steps much larger than the natural time constant of the neutronics.[28]

The application of similar methods to transport theory received attention in the 1960's, but not much practical application. Given the computer speeds of the day, simple linear-difference techniques using more space intervals were generally better. Since problems tended to be small, sufficient memory to store the additional variables was generally available. The economic difficulties lay in the cost of evaluating the various exponential functions required for the analytical solutions. Where a reactor kinetics problem required hundreds of these evaluations, even a small transport problem required millions.

Attempts to use library exponential subroutines were not practical. The library routines were too expensive, and they gave numerical trouble in small intervals, where roundoff became significant. Polynomial approximations were too expensive and/or not sufficiently accurate. Better computers and better mathematics changed all that, however, and analytical solutions across an interval are now very practical.

In search of some insight into this approach, let us first consider a purely numerical method, a fine-grid-coarse-grid formulation. We use Eq. (3.3) in the case of XYZ geometry, where:

$$A_{i+1/2} = A_{i-1/2} = A \quad , \tag{3.47}$$

while $B$ and $C$ are similarly simplified:

$$\begin{aligned} \mu A \left(N_{i+1/2} - N_{i-1/2}\right) + \xi B \left(N_{j+1/2} - N_{j-1/2}\right) \\ + \eta C \left(N_{k+1/2} - N_{k-1/2}\right) + \sigma^T V N = V S \quad . \end{aligned} \tag{3.48}$$

Although this equation is written for a mesh cell as a whole, it would apply equally well to a finer grid along the $i$ direction placed inside the cell. Denoting this grid as $i'$, one has:

$$\begin{aligned} \mu A \left(N_{i'+1/2} - N_{i'-1/2}\right) + \xi B \left(N_{i',j+1/2} - N_{i',j-1/2}\right) \\ + \eta C \left(N_{i',k+1/2} - N_{i',k-1/2}\right) + \sigma^T V N_{i'} = V S_{i'} \quad . \end{aligned} \tag{3.49}$$

Suppose suitable guesses for $N$ at the $j$ and $k$ faces can be found from previous iterations of some process. Since their variation in this context is only with $i'$, we have:

$$\mu A\left(N_{i'+1/2} - N_{i'-1/2}\right) + \sigma^T V N_{i'} = VS_{i'} - \left[ \ldots \right] \equiv VG_{i'} \quad . \tag{3.50}$$

This can be solved for $N_{i+1/2}$, and then Eq. (3.48) yields the average flux, $N$, for the coarse mesh.

With a properly chosen grid, the result would be free of the over-extrapolation problems that plagued previous methods. Yet, the work required for the flux solution, where each coarse mesh interval is divided into $M$ fine meshes, expands only as $M$, rather than $M^3$. The impact on memory requirement would be similarly modest. The spatial variation of flux across the cell volume could be approximated as a linear expansion in the spatial variables and stored for use in calculating subsequent energy groups. Computer codes have actually been constructed to perform such calculations in 2-D. In principle, the implicitness involved in the coupling terms, e.g., $VG_{i'}$ in Eq. (3.50), could be removed by iteration.

This method is not without problems, however. The coding necessary to manipulate the two grids is not simple. If the cells were sufficiently large, the fine-grid calculation would be subject to the same extrapolation difficulties as the coarse-grid method. Although the method allows a spatial distribution of flux inside the cell, its ability to resolve the effects of source and boundary-flux shapes is limited. Accordingly, it would be most useful when most of the flux variation occurs in one dimension.

The semi-analytical methods accomplish a similar solution somewhat more analytically. In the limit of very small $i'$ intervals, Eq. (3.50) approaches a form of the transport equation given by Davison:[10]

$$\mu \frac{d}{dx} N(x) + \sigma^T N = g(x) \quad . \tag{3.51}$$

This is a first-order, linear equation, solvable by integrating factor:

$$N(x) = e^{-\sigma^T x/\mu} \left[ N_{i-1/2}\, e^{\sigma^T x_{i-1/2}/\mu} + \frac{1}{\mu} \int_{x_{i-1/2}}^{x} e^{\sigma^T x'/\mu}\, g(x')dx' \right] \quad . \tag{3.52}$$

Substituting:

$$s \equiv \sigma^T\left(x - x_{i-1/2}\right)/\mu \quad , \tag{3.53}$$

this becomes:

$$N(s) = N_{i-1/2}\, e^{-s} + \frac{e^{-s}}{\sigma^T} \int_0^s e^{s'}\, g(s')\, ds' \quad . \tag{3.54}$$

If $g(s)$ can be expressed as a finite power series:

$$g(s) = \sigma^T \left( g_0 + g_1\, s + \ldots + g_n\, s^n \right) \quad , \tag{3.55}$$

then the solution can be written explicitly in terms of the $C_n$ functions:

$$N = N_{i-1/2}\, e^{-s} + g_0 s C_0(s) + g_1 s^2 C_1(s) + \ldots + g_n s^{n+1} C_n(s) \quad , \tag{3.56}$$

as explained in Appendix F. The $C_n$'s are exponential expressions for which the appendix derives a recursion. Our $C_n$'s differ slightly from Cohen's; Cohen's $C_{n+1}(-x)$ is identical to our $C_n(x)$.[27] Solutions of this type are positive, given positive inputs, and their accuracy holds for very large meshes, subject to the limitations of the expansion of $g(x)$. Equation (3.54) will be the basis of the methods to follow in this section.

The first semi-analytical method discussed here, called the "linear nodal" method, follows an approach pioneered by Walters and O'Dell of Los Alamos.[29-31] To cope with the expense and roundoff problems incurred in using library exponential routines, they found that the Padé(2,3) approximation could give very satisfactory results. The accuracy of this approximation can be seen from Table 3.3. Subsequent work at Oak Ridge[32,33] led to even better approximations for this specific application.

Table 3.3. Accuracy of the Padé(2,3) Approximation to the Exponential

| s | Exact $e^{-x}$ | Padé(2,3) Approximation* |
|---|---|---|
| 0 | 1 | 1 |
| 1/2 | .61 | .61 |
| 1 | .37 | .37 |
| 1 1/2 | .22 | .22 |
| 2 | .14 | .14 |
| 3 | .050 | .054 |
| 4 | .018 | .029 |
| ∞ | 0 | 0 |

$$*e^{-x} \simeq \frac{60 - 24x + 3x^2}{60 + 36x + 9x^2 + x^3}$$

In XYZ geometry, an equation such as Eq. (3.13), with terms with $N_{m+1/2}$ and $N_{m-1/2}$ set to 0, is applied to each cell. The source inside the cell is expanded as a linear function of the space variables:

$$S(X,Y,Z) = S_0 + 2S_X \frac{(X-X_i)}{\Delta X} + 2S_Y \frac{(Y-Y_j)}{\Delta Y} + 2S_Z \frac{(Z-Z_k)}{\Delta Z} \quad , \tag{3.57}$$

and the flux along the incoming cell faces is expanded in forms such as

$$\psi_B(X,Y) = \psi_{B0} + 2\psi_{BX} \frac{(X-X_i)}{\Delta X} + 2\psi_{BY} \frac{(Y-Y_j)}{\Delta Y} \quad , \tag{3.58}$$

providing a boundary source for the new cell. The flux solution inside the cell is assumed to be expandable in the same spatial functions as were used for the source. Writing integrals of the basic equation multiplied, in turn, by each of the space moments, results in a set of coupled 1-D equations, one for each moment.

The resulting equations can be rearranged in the form of Eq. (3.51), and solutions analogous to Eq. (3.54) follow directly. Since one equation is obtained for each expansion parameter, a total of seven equations in 2-D and 13 in 3-D, it is important that the equations be coupled such that an explicit solution is possible. This has been possible in the forms used by TORT. It has also been possible to arrange the equations such that they are correct in the limit of small or zero cross section — an important matter for many of the problems for which the code is intended.

This semi-analytical method, given positive, constant boundary fluxes and sources, provides positive results free of the overextrapolation seen in previous models. The intent of the advanced methods, however, is to allow much larger cell sizes in exchange for more computational work, and the constant-constant assumption proves too limiting. Returning to the full expansions of Eqs. (3.57) and (3.58), we see that, unfortunately, the finite expansion can lead to overextrapolation, resulting in negative or inaccurately positive local sources. Of these, the negatives cause the most harm due to their interference with the convergence process. It happens that repairing the negatives also tends to limit the generation of false positives. One obvious repair is to limit the parameters, e.g.:

$$|S_X| \le S_0; \quad |S_Y| \le S_0; \quad |S_Z| \le S_0; \quad |\psi_{BX}| \le \psi_{B0}, \text{ etc.}, \tag{3.59}$$

but this was not very satisfactory. Another method used at Oak Ridge limited each source parameter separately such that its contribution to each average boundary outflow would be non-negative. This procedure does not quite guarantee non-negativity of the result, however, since two or more parameters, acting together, could still drive the result negative. A more-rigorous approach ensured

that all of the parameters, acting together, could not produce a negative. Both of these methods continue in use and produce equivalent results except in especially difficult circumstances. The first is known as "aggressive nodal", and the second is "safe nodal". These procedures add significantly to the computational work, but they vectorize well.

Another, more obscure, procedure was required to prevent anomalous behavior when small regions adjoin large ones. The result of the nodal procedure is an equation set similar to Eq. (3.34) in form, and it is possible to identify expressions that play the role of the weighting coefficients. In the nodal method, however, these tend to vary from the linear limit, in the context of weighted difference, past the step limit, and onward to a "flat flux" limit. This last regime was found to accompany the anomalous behavior, and limiting the weighting expression to the step limit repaired the trouble. This limit also vectorizes, and it is a necessary part of the TORT procedure.

The references demonstrate the superiority of the nodal method in problems such as the penetration of neutrons through thick regions of shield material. One of these problems shows that the method also works well through void regions. The additional computation was generally more than offset by the improved accuracy.

One of the important applications of a 3-D code such as TORT is to calculate the penetration of gammas into large open areas, such as building or room interiors. These particles behave less like gas molecules, bouncing about to fill the space, and more like visible light photons, scattering much less, and giving areas of high illumination and deep shadow.

In such a problem, it is important that the numerical method not fill the shadow areas with particles misdirected by the limitations of the method. Such artificial redirection has taken on the name "lateral dispersion." Let us examine a simple case shown in Fig. 3.2 (pg. 3-54), in which a monodirectional source of magnitude 1.0 flows past a sharp-edged, opaque obstruction. It happens that this is an easy case to solve in 2-D if the mesh cells are equilateral and the $S_2$ approximation is used. The correct result is, of course, a flux of 1.0 along the face *CD* and 0.0 along face *BC*. Both weighted difference and the linear method can solve this problem without difficulty. Both the simple step method and the sophisticated nodal method force a portion of the particles across the *BC* boundary and into the shadow area, where they should not be. Appendix G discusses dispersion results and difficulties in more detail.

Considerations such as this led to a study of characteristic methods for TORT. At the expense of additional complexity, they can combine insensitivity to large mesh widths with a rigorous calculation of flux distribution along the cell boundaries. Looking at Eq. (3.54), it is evident that integration in the variable *s* is equivalent to integration along actual particle paths. In fact, solution along paths parallel to *AC* in the area *ACD* produces the exact solution along the boundary *CD*. Spatial integrals yield the average cell flux and average emergent boundary fluxes. Additional relationships yield higher spatial moments of flux inside the cell and on the faces for use in performing scattering integrals and coupling to other cells. The characteristic method combines

positive results, good resistance to dispersion, and excellent bulk-penetration capability. Formal discussions and studies of characteristic methods have been reported by Lathrop,[23] Alcouffe and Larsen,[34] Larsen and Alcouffe,[35] and elsewhere.

In general, one must suspect that there is a strong relationship between nodal and characteristic methods, since both amount to applications of Eq. (3.54) across the cell volume. Beernink and Dorning[36] examined this relationship and found that the difference lies largely in the treatment of boundary fluxes as adjacent zones are coupled, the very matter that led us to consider the characteristic method. Since the exact distribution along the boundary is available, the method of coupling can be devised to suit a given class of applications.

Several options were considered for TORT. One can calculate the higher spatial moments of the flux quite rigorously from suitable spatial integrals, but this leads to a relatively expensive equation set. Another method projected the centered slope onto the boundary; in the example of Fig. 3.2, the slope of flux along *CD* would be found from the cell-averaged flux, the average flux across BC, and the distance 1/2 $\Delta Y$. While this method worked well for the calculation of reactor eigenvalues, it was not satisfactory for deep-penetration work.

Finally, a semi-intuitive method gave broad applicability and robustness. In this, source and flux inside the cell are expanded in separable functions of the space variables:

$$S(X,Y,Z) = S_0\left[1 + 2S_X \frac{(X-X_i)}{\Delta X}\right]\left[1 + 2S_Y \frac{(Y-Y_j)}{\Delta Y}\right]\left[1 + 2S_Z \frac{(Z-Z_k)}{\Delta Z}\right] , \qquad (3.60)$$

where $\Delta X$, $\Delta Y$, and $\Delta Z$ are the cell widths, and fluxes at the cell faces are similarly expanded in functions such as, for the left face:

$$\psi_L(X,Y) = \psi_{L,0}\left[1 + 2\psi_{L,Y} \frac{(Y-Y_j)}{\Delta Y}\right]\left[1 + 2\psi_{L,Z} \frac{(Z-Z_k)}{\Delta Z}\right] , \qquad (3.61)$$

Referring to Fig. 3.2 once again, the slope coefficients inside the cell are determined from average boundary values, e.g.:

$$S_x = \frac{\psi_{CD} - \psi_{AB}}{2\psi_0} , \qquad (3.62)$$

where $\psi_{AB}$ and $\psi_{CD}$ are average fluxes along faces *AB* and *CD* and $\psi_0$ is the cell-average flux.

The fluxes at corners *B* and *D* in the 2-D example of Fig. 3.2 are known from the conditions placed across faces *AB* and *AD* by the coupling from adjacent cells. Once the average emergent fluxes

across faces $BC$ and $CD$ are known, the flux, $\psi_C$, at corner $C$ is estimated, assuming separable spatial shapes once again:

$$\psi_C = \psi_0 \left( \frac{\psi_{BC}}{\psi_0} \right) \left( \frac{\psi_{CB}}{\psi_0} \right) , \tag{3.63}$$

where $\psi_{BC}$ is the average flux along $BC$. Then, the slope parameters along the boundaries are taken as, for example, in the case of the right boundary:

$$\psi_{R,Y} = \frac{\psi_C - \psi_B}{2\psi_{BC}} , \tag{3.64}$$

where $\psi_B$ is the flux at $B$.

Both internal and face parameters must be limited to prevent Eqs. (3.60) and (3.61) from implying negatives, but this limiting has proven relatively benign. The method has shown good convergence behavior in mesh-refinement studies, and it gives plausible results in dispersion situations. It is apparent that the flexibility of the characteristic formulation also allows other coupling procedures to be tried in the future.

### 3.2.7 Choosing the Proper Method

After the previous section, the user may feel the need for guidance as to the proper choice of method. Historically, the development and acceptance of each method for general use has been approximately in the order of increasing complexity — unfixed linear, linear-step, linear-zero, weighted, nodal, and characteristic. Both weighted and characteristic methods were known in the 1960's, but widespread use did not immediately follow their introduction. It will be seen that the proper choice depends upon cost, numerical performance, and the type of problem being solved.

The use of non-positive methods such as the unfixed linear method is generally unacceptable in work for which TORT is intended. While such a method may produce good global eigenvalues in a finely–meshed reactor problem, the inevitable negatives found in general transport calculations cause unacceptable local errors and thwart convergence. In a 1969 review article, Lathrop[23] suggested that the linear-zero method be used for the calculation of global parameters, e.g., reactor eigenvalues, and that weighted difference be used otherwise. This is still good general advice many years later, at least with respect to those methods.

Linear-zero calculations are relatively economical, and they are highly trusted in such applications. Although zero is generally an underestimate of the true flux, this is not a matter of great concern for reactor calculations. Weighted difference can also be used in reactor problems, but a value of the theta parameter should be 0.3 or greater. Linear-zero is probably more commonly used and trusted in this field, however. It is not clear whether this is due to better accuracy or to historical

happenstance. Satisfactory comparisons, especially in 3-D, are lacking. The use of the zero--weighted procedure in such cases leads to large errors, however.[37]

Coding technique and computer architecture may play an important role in the future balance between linear-zero and theta-weighted efficiency. The zero fixup has been implemented in both 1-D and 2-D by coding tests and separate procedures for each possible combination of negative results. The increase in complexity from 1-D to 2-D was impressive, however. The extension of that approach to 3-D would not appear practical, and alternate procedures are needed.

On the other hand, the extension of weighted difference to 3-D is a simple matter. The calculation of all but one of the weights can be done with vector instructions. The final weight calculation and recursion sweep involves only a few statements that have been coded in an assembler language.

The linear-zero method can be used in ex-core radiation transport work, although it is not so advantageous there, and the tendency to underestimate flux can lead to non-conservative results that are sometimes not acceptable. A 1979 comparison[21] demonstrated large differences between linear-zero and weighted results in deep-penetration problems prepared according to conventional practice. Although the linear-zero method required less computation, the weighted method was generally more economical due to faster, smoother convergence. In general, it is the weighted results that are trusted and compared to experimental results in this field.

If weighted difference is to be chosen, what value of theta should be used? While the cost does not vary much with theta, the results vary significantly. Thetas of 0.5 and greater generally give much better performance in mesh-refinement studies and better agreement with the linear-fixup methods.[17,33] The dependence on theta in this range is weak, and it is not intended that an optimum value for general use should be sought.

The zero-weighted method has its advocates, however. The linear-zero method can produce non-physical ripples in flux shape, and those ripples have been found, to a lesser extent, in theta-weighted results.[17] Zero-weighted results do not have these ripples, and they do have a history of successful comparison with certain classes of experiment.[38] Whether this is fortuitous or not appears impossible to prove.

In summary, then, one can use linear-zero procedures or theta-weighted procedures with confidence for reactor core calculations. Theta-weighted procedures are the best available for ex-core radiation transport in curved geometry. In XYZ geometry, characteristic procedures appear to offer the best accuracy where positive results are required. The performance of the nodal procedures is generally well accepted, however.

There has not been a great amount of testing of nodal and characteristic procedures in the area of reactor eigenvalue calculations. One might suspect that they might do well, but performance over

a wide range of problems would have to be proven. The lack of cylindrical geometry would be a limitation in some cases.

The issue of linearity, in the mathematical sense, is of concern to some. All of the procedures are homogeneous operations of degree 1 with respect to a single positive source; i.e., if $F$ is the solution corresponding to source $S$, then $bF$ is the solution corresponding to the source $bS$ if $b$ is non-negative. Superposition does not necessarily apply, however. If $G$ is a solution corresponding to source $T$, then $F+G$ will not necessarily be the solution for source $S+T$ if negative prevention is employed. Nonlinear methods have certain disadvantages. They tend to be more troublesome to converge. The user cannot solve problems legitimately having negative sources. If a source is broken into components, the user cannot be confident that the sum of the solutions will add to the solution for the original source. There is difficulty in obtaining correct forward-adjoint folding results.

Of the methods discussed here, only the unfixed linear method, the step method, and the zero-weighted method used in DOT III operate without some kind of test that depends upon the computed values, however, and these tests thwart superposition. The nodal and characteristic methods are linear in first principal, but the negative-prevention procedures result in non-linearity. Accordingly, truly linear procedures do not find much practical use.

### 3.2.8 The Source Term

The source term in Eq. (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

$$S_{i,j,k,m,g} = Q_{i,j,k,m,g} + \chi_{i,j,k,g} \sum_{g'} \nu_{i,j,k,g'} \sigma^{F}_{i,j,k,g'} \sum_{m'} W_{m'} N_{i,j,k,m',g'} + \sum_{g'} \sum_{m'} W_{m'} \sigma^{S}_{i,j,k,m,g,m',g'} N_{i,j,k,m',g'} \quad . \tag{3.65}$$

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, it can be argued that the cross sections might vary in a pointwise manner due to local temperature or density effects or to local differences in the spectral weighting used in obtaining group cross sections. In practice, however, such detail is rarely required; the detailed pointwise description is typically not available, and the mass of data would be difficult to generate and store. Local temperature and weighting effects can be represented by defining enough zones so that only cells of acceptably similar properties are grouped together. In the discussion to follow, the subscript $z$ will be used to denote material zone, with the understanding that the zone number is a function of $(i,j,k)$.

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

$$D^F_{i,j,k} = \sum_{g'} \nu\sigma^F_{z,g'} \bar{N}_{i,j,k,g'} \quad , \tag{3.66}$$

and

$$\bar{N}_{i,j,k,g'} = \sum_{m'} W_{m'} N_{i,j,k,m',g'} \quad . \tag{3.67}$$

$D^F_{i,j,k}$ is termed "fission density," while $\bar{N}$ is "scalar flux." If several fissile species are present in the system, the use of a single $\chi_g$ 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. It could be changed readily if future needs demand it.

In principle, Eq. (3.65) can be used directly to calculate scattering. If, as discussed, zone cross sections are used, the scattering term is:

$$\sum_{g'} \sum_{m'} W_{m'} \sigma^S_{z,m,g,m',g'} N_{i,j,k,m',g'} \quad . \tag{3.68}$$

In fact, certain transport studies have been performed in this way. In TORT however, as in most contemporary discrete ordinates codes, a moment expansion, as discussed by Mynatt,[15] is used.

The directional dependence of $\sigma^S$ in Eq. (3.1) is assumed to be represented satisfactorily by a Legendre polynomial in the single variable $\mu_0$, where:

$$\mu_0 = \underline{\Omega} \cdot \underline{\Omega}' \quad , \tag{3.69}$$

so that:

$$\sigma^S\left(\underline{r}, E, E', \underline{\Omega}, \underline{\Omega}'\right) = \frac{1}{4\pi} \sum_{l=0}^{l^{max}} P_l\left(\mu_0\right) \sigma^S_l\left(\underline{r}, E, E'\right) \quad . \tag{3.70}$$

In Eq. (3.65), the corresponding representation is:

$$\sigma^S_{z,m,g,m',g'} \simeq \frac{1}{4\pi} \sum_{l=0}^{l^{max}} P_l(\mu_0)\sigma^S_{l,z,g,g'} \quad , \tag{3.71}$$

where $\mu_0$ is now the cosine of the angle between directions $m$ and $m'$. Since the $\sigma^S_l$ 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. (3.71), which results in values of $\sigma^S_l$ that 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 ORNL codes.

As discussed below, it is fruitful to apply a corresponding expansion to $\psi$. Since $\psi$ is inherently a function of the full direction space, a type of spherical harmonic expansion is employed where $\psi$ is approximated as:

$$\psi(\underline{r},E,\Omega) \simeq \sum_{l=0}^{l^{max}} \sum_{k=0}^{2l} (2l+1)\, C_l^k(\mu,\phi)\psi_l^k(\underline{r},E) \quad , \tag{3.72}$$

with:

$$C_l^k(\mu,\phi) \equiv \left[\frac{(2-\delta_0^K)(l-K)!}{(l+K)!}\right]^{1/2} P_l^K(\mu)\, f^K(\phi) \quad , \tag{3.73}$$

and:

$$f^K(\phi) \equiv \cos K\phi;\ K \text{ even or } 0$$
$$\equiv \sin K\phi;\ K \text{ odd},$$

where $K = k/2$ if $k$ even, or $K = \frac{k+1}{2}$ if $k$ odd, and where $\delta_0^K$ is the "Kronecker delta," 1 when $K$=0 and 0 otherwise, and the $P_l^K$ are the associated Legendre polynomial basis functions. In XYZ geometry, if $\underline{i}$, $\underline{j}$, and $\underline{k}$ are unit vectors along the X, Y, and Z coordinate axes, $\phi$ is the angle between the planes formed by the $\Omega$ and $\underline{k}$ vectors and by the $\underline{k}$ and $\underline{i}$ vectors. (In R$\Theta$Z geometry, $\underline{i}$ and $\underline{j}$ are vectors along the radial and azimuthal coordinate axes.)

On the set of discrete directions {m}, the function $C_l^k$ is represented by an array of discrete values:

$$C^k_{l,m} \equiv C_l^k(\mu_m,\phi_m) \quad , \tag{3.74}$$

and the corresponding flux approximation is:

$$N_{i,j,k,m,g'} \simeq \sum_{l=0}^{l^{max}} \sum_{k=0}^{2l} C_{l,m}^{k} N_{l,i,j,k,g'}^{k} \quad . \tag{3.75}$$

The advantage of this is twofold: The number of moments is generally much less than the number of directions. Also, it can be shown that the scattering portion of the source term can be obtained directly from the moments:

$$\sum_{g'} \sum_{m'} W_{m'} \sigma_{i,j,m,g,m',g'}^{S} N_{i,j,k,m',g'} \simeq \sum_{g'} \sum_{l} \sum_{k} \sigma_{l,z,g,g'}^{S} N_{l,i,j,k,g'}^{k} \quad . \tag{3.76}$$

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

$$Q_{i,j,k,m,g} \simeq \sum_{l=0}^{l^{max}} \sum_{k=0}^{2l} C_{l,m}^{k} Q_{l,i,j,k,g}^{k} \quad , \tag{3.77}$$

and it is called a "distributed source" in this form. The source in a given direction is easily found using:

$$S_{i,j,k,m,g} = \chi_g D_{i,j,k} + \sum_{l=0}^{l^{max}} \sum_{k=0}^{2l} C_{l,m}^{k} \left[ Q_{l,i,j,k,g}^{k} + \sum_{g'} \sigma_{l,z,g,g'}^{S} N_{l,i,j,k,g'}^{k} \right] \quad . \tag{3.78}$$

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

$$N_{l,i,j,k,g}^{k} = \sum_{m} W_m C_{l,m}^{k} N_{i,j,k,m,g} \tag{3.79}$$

and stored. This allows the $N_m$'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. (3.75) 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. (3.78) will not be correct. On the other hand, the theory that allows evaluation of the $N_l^k$ by Eq. (3.79) assumes that integration over all direction space will be performed, in which case the $C_l^k$ are orthogonal. When the discrete quadrature of Eq. (3.79) is used, the corresponding orthogonality condition is satisfied only when $l^{max}$ is sufficiently small. Finally, the memory required to store the moments expands roughly as $l^{max}$ squared, so that higher-order expansions can be quite burdensome.

In addition, the cross section expansion of Eq. (3.70) can badly misrepresent certain cross sections, resulting in spurious oscillation between positive and negative scatters. Cross sections representing high-energy Compton scattering are an example of this.

In spite of these difficulties, the "$P_l\ S_n$" method described here has proven generally successful. Codes are available to test the orthogonality of the basis functions over specific quadrature sets. Procedures are now available to mitigate the effects of the spurious oscillations. Cross section representation has been accurate enough to give good comparisons with experiment. The method is now widely successful.

## 3.3 PROBLEM SPECIFICATION

### 3.3.1 Directional Quadrature Sets

As noted earlier, integrals over direction space are approximated by 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 $(\mu,\xi,\eta)$ with respect to a set of orthogonal coordinate axes. With $\underline{i}$, $\underline{j}$, and $\underline{k}$ as unit vectors along the axes, the discrete direction set is

$$\mu_m \underline{i} + \xi_m \underline{j} + \eta_m \underline{k} \quad . \tag{3.80}$$

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:

$$\frac{1}{4\pi} \int f(\underline{\Omega})\, d\underline{\Omega} \simeq \sum_m W_m f\left(\mu_m, \xi_m, \eta_m\right) \quad . \tag{3.81}$$

One constraint on the sets is immediately apparent:

$$\mu_m^2 + \xi_m^2 + \eta_m^2 = 1 \quad . \tag{3.82}$$

It is also required that the zero-th directional moment be 1 and that the first-order moments of a symmetrical function to be zero. This ensures that isotropic flux will have a magnitude equal to the scalar flux, but zero current:

$$\sum W_m = 1 \quad , \tag{3.83}$$

$$\sum W_m \mu_m = 0 \quad , \tag{3.84a}$$

$$\sum W_m \xi_m = 0 \quad , \tag{3.84b}$$

and

$$\sum W_m \eta_m = 0 \quad , \tag{3.84c}$$

These constraints are sufficient to specify the simplest practical direction set, a fully symmetrical $S_2$. This direction set has one direction in each octant, eight in all. By full symmetry, we mean that, if $(\mu_m, \xi_m, \eta_m)$ is in the set, then $(\xi_m, \eta_m, \mu_m)$ and $(\eta_m, \mu_m, \xi_m)$ are also in the set. It can be seen that this symmetry is obtained if:

$$|\mu_m| = |\xi_m| = |\eta_m| = \delta \text{ for all } m \quad . \tag{3.85}$$

From Eq. (3.82), we see that $\delta = 1/\sqrt{3}$ . The eight combinations of $\pm\delta$, taken three at a time, give the required eight directions. The remaining constraints of Eqs. (3.83) and (3.84) are satisfied by:

$$W_m = 1/8 \quad . \tag{3.86}$$

As the number of directions increases, the larger number of unknowns allows additional constraints to be added in order to provide maximum accuracy. Jenal *et al.*[39] discuss this matter in more detail. Their 2-D discussion is also applicable to 3-D.

Proper ordering of the directions is essential to the iteration process. All directions with common $\eta$ must be placed together in groups called "$\eta$-levels." All $\eta$-levels with $\eta<0$ must precede all directions with $\eta>0$. Within an $\eta$-level, directions with $\xi \leq 0$ are ordered according to increasing $\mu$. This direction sequence is then repeated as the code will supply negative $\xi$'s for the first sequence and positive $\xi$'s for the second.

If zero-weight initiating directions are used, each $\eta$-level must have, as its first member, an arbitrary boundary direction for which $\xi=0$. Thus, for a given $\eta$, the boundary direction has:

$$\mu = -\sqrt{1-\eta^2} \quad . \tag{3.87}$$

In the $S_2$ example given above, 4 new directions would be required:

$$\left(\pm\delta^0,\ 0,\ \pm\delta\right) \quad , \tag{3.88}$$

where

$$\delta^0 = \sqrt{1-\delta^2} \quad . \tag{3.89}$$

These zero-weight directions are used in curved geometry to start the sweep through adjacent directions in an $\eta$-level. They are unneeded and unused in XYZ geometry. In fact, Lathrop and Brinkley[16] use initiating directions with non-zero weights in all geometries to good effect. Tomlinson *et al.* showed that zero-weight or non-zero weight initiating directions should be treated with the same fundamental assumptions. With that provision, either gave good results for the problems studied. One can speculate that sets with non-zero initiating directions could be more efficient. They use fewer directions for a given "order" of direction set, i.e., for a given number of $\eta$-levels. A definitive proof of this idea is not at hand, however.

Sets can be "biased" or "asymmetric" in $\eta$, e.g., additional directions can be supplied along the -Z direction in order to give fine detail to polar streaming. Asymmetry in $\mu$ can also be accommodated, although such applications have been rare.

### 3.3.2 External Boundary Conditions and Sources

In a previous section, it was noted that the value of $N$ must be artificially set at each boundary 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 zero.

The other common condition is "reflected." This option describes the condition at a plane of symmetry, such as 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 that 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 $(\mu,\xi,-\eta)$ is set equal to the flux emerging in direction $(\mu,\xi,\eta)$. At a vertical right-hand boundary, flux entering in direction $(-\mu,\xi,\eta)$ for example, is set to the flux emerging in direction $(\mu,\xi,\eta)$.

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

$$N_m = \alpha \frac{\sum_{m'} W_{m'} N_{m'} \; ; \; \eta_{m'} > 0}{\sum_{m'} W_{m'} \; ; \; \eta_{m'} < 0} \quad . \tag{3.90}$$

This condition is sometimes used in shielding work and air transport problems to give an approximate representation of material for which a detailed calculation is not required. Its most serious limitation is that it can not treat the change in energy that is likely to accompany such a reflection. If the incident flux is anisotropic, the directional distribution of the emerging flux is also overlooked. A "white" boundary condition is simply the special case of $\alpha$=1. It is sometimes used at the outer boundary of cells, although the cylindrical condition, to be described next, is better.

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

$$N_m = \frac{\sum_{m'} W_{m'} N_{m'} \; ; \; \mu_{m'} \, , \, \mu_m < 0 \; ; \; \eta_{m'} = \eta_m}{\sum_{m'} W_{m'} \; ; \; \mu_{m'} \, , \, \mu_m > 0 \; ; \; \eta_{m'} = \eta_m} \quad . \tag{3.91}$$

This is much better than any of the preceding conditions at the outer boundary of a 2-D cylinder representing one cell of a regular lattice.[40] It is also more appropriate at the left edge of an RZ mesh, since it ensures that outward flux will be truly independent of $\mu$. None of the conditions described here are truly correct at the center of a cylinder that is not azimuthally symmetrical, however.

When applied at a horizontal boundary, the cylindrical condition allows a degree of asymmetry in $\mu$:

$$N_m = \frac{\sum_{m'} W_{m'} N_{m'} \; ; \; \eta_{m'} = -\eta_m}{\sum_{m'} W_{m'} \; ; \; \eta_{m'} = \eta_m} \quad . \tag{3.92}$$

In theory, this allows reflection at a horizontal boundary at which the $\mu$'s for given $\eta$ pairs do not match. This application has not received much practical use, however.

The "periodic" boundary condition represents a continuing array of asymmetric zones, 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 $i=I+1/2$ is the right boundary opposite the left boundary $i=1/2$:

$$N_{1/2} = N_{I+1/2} \quad . \tag{3.93}$$

The periodic boundary condition is the correct condition for the inside and outside boundaries in RΘZ geometry, i.e. Θ boundaries.

The "fixed-source" or "fixed-flux" boundary condition simply sets the flux to a value specified as input. It is most often used in problems that are spatial continuations or refinements of previous problems. Peripheral codes allow flux from a previous 2-D or 3-D problem to be used as fixed-flux boundary conditions, thus allowing "bootstrapping."

### 3.3.3 Internal Boundary Sources

For certain applications, it is desirable to have a source comparable to the external boundary source described above, but located at an internal boundary. If we represent, for example, a source at the $i+1/2$ boundary as $S^{B}_{i+1/2,j,k,m,g}$, then the effect of the source is to make the flux leaving the boundary larger than the flux entering the boundary by the amount $S^{B}$. For example, if the sweep is from small $i$ to large, and if the overflow from the cell preceding boundary $i+1/2$ in the direction of sweep is $N_{i+1/2}$, then the flow into cell $i'$, where $i'=i+1$, is:

$$N_{i'-1/2} = N_{i+1/2} + S^{B}_{i+1/2} \quad . \tag{3.94}$$

### 3.3.4 Units

Either standard SI units or the more traditional barn-centimeter-Angstrom units can be used. Results can be instantaneous or time-integrated. With sources per unit time, the results are fluxes, absorption rates, etc. With sources integrated over the desired time span, the results are fluences, total absorptions, etc. Table 3.4 illustrates both SI and traditional units for the case of sources per unit time. For time-integral sources, simply ignore the references to seconds.

Since the directional weights sum to 1, they are unitless. Integrals over direction are replaced by weighted sums. Scalar fluxes, $\phi$'s, are related to directional fluxes as follows:

$$\phi = \sum W_m \psi_m \quad . \tag{3.95}$$

From this, it is evident that $\phi$ and $\psi$, must, therefore, have the same units. By dividing $\psi$, as calculated by TORT, by $4\pi$ steradians per unit sphere, units of flux per steradian (sr) can be obtained. Also, by multiplying $\theta$ by $2\pi$, units of radians (rad) can be obtained. These units are not acceptable for use in the code, however.

In the optional 2-D geometries, the "missing" dimension is entirely ignored. Thus, the results are normalized as if the system were of unity size in the missing dimension.

Table 3.4. Unit Systems

| | SI Units | Traditional Units |
|---|---|---|
| Space dimensions: R, Z, X, Y | Meters (m) | Centimeters (cm) |
| Time | Seconds (s) | Seconds (s) |
| Rotation fraction, $\Theta$ | Dimensionless | Dimensionless |
| Weight, $W$ | Dimensionless | Dimensionless |
| Atomic densities | m-3 | Angstrom$^{-3}$ |
| Micro cross sections | $m^2$ | barns |
| Macro cross sections | $m^{-1}$ | $cm^{-1}$ |
| Scalar fluxes | $m^{-2}s^{-1}$ | $cm^{-2}s^{-1}$ |
| Directional fluxes | " | " |
| Boundary sources | " | " |
| Distributed sources | $m^{-3}s^{-1}$ | $cm^{-3}s^{-1}$ |

## 3.4 SOLUTION PROCEDURES

### 3.4.1 Iteration Strategy

Two of the chief advantages of discrete ordinates theory over diffusion theory 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 upon 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 "flux" or "inner" iterations may typically be required. 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, outer, or right-hand boundary of a mesh also require iterations, as do periodic boundary conditions at any boundary.

Problems that 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 is 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 resolved by estimating the fission and upscatter contributions from previous information or from the input flux guess, and by 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 a small number of flux evaluations at each pass through the groups if several source iterations are to be performed. Studies have shown that 2 or 4 flux iterations per source iteration give optimum results for certain problems. An even number of iterations tends to give quicker convergence than an odd number. The number of source iterations required may vary from as few as 4 to 100 or more, depending upon the problem, the success of the acceleration procedures, and the degree of convergence required.

### 3.4.2 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 $l=3$ often produces troublesome negative and false positive scattering, which thwarts convergence and renders 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 cross sections in such cases has shown that the fitting procedure often results in good fits where the cross section is large, with meaningless oscillation between small positive and negative errors elsewhere. General repairs for this difficulty have been developed,[41] but none are in widespread use, due to cost, complexity, or other limitations.

This code allows, as an option, a choice of non-rigorous repairs that require no special data, and that have generally controlled the undesirable expansion effects. The first method preserves the total source magnitude at each point. 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,[42] it usually improves convergence, and has been used frequently in radiation transport work. If few negatives are found, the effect on flux solution time can be as little as 15%. Additional economy can be gained by abandoning the search for negatives after a group of higher energy has been completed without finding any. Experience has indicated that no further negatives are found, in typical cases, after such a group. In two-particle problems, the search must be resumed for the first group of the second particle type, of course. If many negatives are found, the cost of removing them can be quite high, and the answers may be changed significantly. Even so, many users will not consider running problems without this repair.

An "economy" method simply sets the negatives to zero. This is at least as beneficial to convergence, and the resulting upper-limit treatment can be preferable in certain applications. The cost of the economy procedure is negligible. It is unsuitable in cases such as $k_{eff}$ calculations, where particle conservation must be rigorous, however.

### 3.4.3 Fission Problems

If a fixed source (distributed, internal boundary, or external boundary) is present, Eqs. (3.3) and (3.65) can be iterated to a stable solution, providing $k_{eff}$ 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 a machine having 4-byte arithmetic. As $k_{eff}$ increases, $M$ approaches $\infty$, and no stable solution can be reached.

Without a fixed source, however, meaningful solutions are obtained by searching for the $k_{eff}$ by which $\chi$ must be divided in order to bring Eqs. (3.3) and (3.65) to a stable solution. This code uses a very simple procedure for the basic evaluation of $k_{eff}$. The initial source is calculated based upon the flux guess or a distributed source initialization. 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 an iteration with extrapolated results which are nearer the ultimate solution.

Problem-modifying search options are available. The absorber search ("*A*") multiplies an input value of an absorber distributed in space and energy by an adjustable parameter (the "eigenvalue," $\lambda$) until the resulting value of $k_{eff}$, evaluated as described earlier, is a specified value. The concentration search ("*C*") adjusts $\lambda$ 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 $\lambda$ such as to find new widths for specified mesh intervals which achieve the desired value of $k_{eff}$.

The search for $\lambda$ is conducted in the same manner in each type of search. The value of $\lambda$ is set to an input value, $\lambda_1$, and then $k_1$, a partially-converged estimate of $k_{eff}$, is obtained. A new value, $\lambda_2$, is based upon user-supplied information and $k_1$, after which $k_2$ is obtained. At this point, a nonlinear procedure that tends to prevent overextrapolation is used to estimate $\lambda_3$, from which $k_3$ is evaluated. From this point onward, the newest three values of $\lambda$ and corresponding estimates of $k$ are used to extrapolate until $k$ is sufficiently near the specified value.

The nonlinear extrapolation uses a fitting function suggested by Vondy.[43] Where *a, b,* and *c* are arbitrary fitting parameters:

$$\lambda = c + ak/(b-k) \quad . \tag{3.96}$$

If only two points are available for the fit, $c$ is arbitrarily assumed to be 0. If a negative $\lambda$ 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 $\lambda$ has reversed signs, i.e., if:

$$\left(\lambda_N - \lambda_{N-1}\right)\left(\lambda_{N-1} - \lambda_{N-2}\right) < 0 \quad , \tag{3.97}$$

then oscillation damping is applied. Where $\tilde{\lambda}_{N+1}$ is the new estimate of $\lambda$ calculated from Eq. (3.97), a modified $\lambda_{N+1}$ is tried:

$$\lambda_{N+1} = \lambda_N + \gamma\left(\tilde{\lambda}_{N+1} - \lambda_N\right) \quad . \tag{3.98}$$

The value of $\gamma$ is presently set at 0.8.

As an additional protection against meaningless extrapolation, $\lambda$ is required to stay within an input factor of its original value. If $k$ departs from the objective value by more than a specified value, or if $\lambda$ is "stuck" on the same value for two successive estimates, the search is halted. The relative change in $\lambda$ per estimate is limited to a specified value.

This is certainly not a foolproof scheme. It will obviously be defeated if the allowable variation in $\lambda$ is not sufficient to adjust $k$ to the desired value. If an input error causes $\lambda$ to have very little effect upon $k$, a perplexing and aimless drift will be observed. The input includes a "convergence ratio", which allows $\lambda$ to change when $k$ is only partially converged. An unduly low value of the ratio can cause premature extrapolation of $\lambda$, and subsequent poor results. In general, however, the scheme has worked well for careful users.

## 3.5 FLUX ITERATION ACCELERATION

### 3.5.1 Groupwise Rebalance

The flux iterations in a difficult problem must be accelerated in some way in order to bring the cost within acceptable limits. The oldest and simplest method is "group rebalance." First, let us split the source term of Eq. (3.65) into two parts, separating the scattering portion that originates within the energy group from the remainder of the source, and call these $H_g$ and $G_g$, respectively:

$$\overline{\overline{H}}_g \equiv \sum_i \sum_j \sum_k V_{i,j,k} \sum_m W_m \sum_{m'} \sigma^S_{m,g,m',g} N_{m',g} \quad , \tag{3.99}$$

$$\overline{\overline{G}}_g \equiv \sum_i \sum_j \sum_k V_{i,j,k} \left[ \chi_g D + \sum_m W_m \left[ Q_{m,g} + \sum_{g' \neq g} \sum_{m'} W_{m'} \sigma^S_{m,g,m',g} N_{m',g} \right] \right] \quad . \tag{3.100}$$

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:

$$\overline{\overline{L}}_g + \overline{\overline{T}}_g = \overline{\overline{G}}_g + \overline{\overline{H}}_g \quad , \tag{3.101}$$

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

$$\tilde{\overline{\overline{L}}}^p + \tilde{\overline{\overline{T}}}^p = \overline{\overline{G}} + \overline{\overline{H}}^{p-1} \tag{3.102}$$

When $H^p$ is calculated from the $p^{th}$ iteration fluxes, it will give rise to different fluxes in the $(p+1)^{th}$ iteration. If $H$ is almost as large as $L+T$, 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. Let us assume an arbitrary factor, $f$, by which all fluxes 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:

$$f\,\tilde{\overline{\overline{L}}}^p + f\,\tilde{\overline{\overline{T}}}^p = \overline{\overline{G}} + f\,\tilde{\overline{\overline{H}}}^p \tag{3.103}$$

Solving Eqs. (3.102) and (3.103) to eliminate $L$ and $T$:

$$f_g = \frac{\overline{\overline{G}}_g}{\overline{\overline{G}}_g - \left(\tilde{\overline{\overline{H}}}_g^p - \overline{\overline{H}}_g^{p-1}\right)} \quad . \tag{3.104}$$

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

$$N_g^p = f_g \tilde{N}_g^p \quad . \tag{3.105}$$

This simple procedure was helpful in early codes used on problems in which flux shape is dominated by leakage, such as small metal assemblies. It is not applicable to large systems, however. Even so, $f$ is used as one indicator of overall convergence.

### 3.5.2 Space Rebalance

By 1968, discrete ordinates codes were being applied to deep-penetration transport problems, and global rebalance was no longer adequate. From similar considerations applied on a mesh-cell basis, "space rebalance" was developed.[44] Integrating Eq. (3.3) over direction, but not over space, we define "partial currents":

$$F_{i+1,i} \equiv A_{i+1/2} \sum_m N_{i+1/2,m}\, \mu_m\, W_m \;\; ; \;\; \mu_m > 0 \;\; ; \tag{3.106a}$$

$$F_{i,i+1} \equiv A_{i+1/2} \sum_m N_{i+1/2,m}\, \mu_m\, W_m \;\; ; \;\; \mu_m < 0 \;\; ; \tag{3.106b}$$

$$F_{j+1,j} \equiv B_{j+1/2} \sum_m N_{j+1/2,m}\, \xi_m\, W_m \;\; ; \;\; \xi_m > 0 \;\; ; \tag{3.106c}$$

$$F_{j,j+1} \equiv B_{j+1/2} \sum_m N_{j+1/2,m}\, \xi_m\, W_m \;\; ; \;\; \xi_m < 0 \;\; ; \tag{3.106d}$$

$$F_{k+1,k} \equiv C_{k+1/2} \sum_m N_{k+1/2,m}\, \eta_m\, W_m \;\; ; \;\; \eta_m > 0 \;\; ; \tag{3.106e}$$

and

$$F_{k,k+1} \equiv C_{k+1/2} \sum_m N_{k+1/2,m}\, \eta_m\, W_m \;\; ; \;\; \eta_m < 0 \;\; ; \tag{3.106f}$$

Similarly, the "inscatter source":

$$\overline{G}_g \equiv \chi_g\, D + \sum_m W_m \left[ Q_{m,g} + \sum_{g' \neq g} \sum_{m'} W_{m'}\, N_{m',g'}\, \sigma^S_{m,g,m',g'} \right] , \tag{3.107}$$

the "scalar self-scatter":

$$\overline{\sigma}^S \equiv \sum_m \sum_{m'} W_{m'}\, W_m\, \sigma^S_{m,g,m',g'} \tag{3.108}$$

and the "removal":

$$\sigma^R \equiv \sigma^T - \bar{\sigma}^S \quad , \tag{3.109}$$

are defined. The integrated result is a scalar balance equation which would be satisfied at convergence.

$$\begin{aligned} & F_{i+1,i} - F_{i,i+1} + F_{i-1,i} - F_{i,i-1} + F_{j+1,j} - F_{j,j+1} + F_{j-1,j} - F_{j,j-1} \\ & + F_{k+1,k} - F_{k,k+1} + F_{k-1,k} - F_{k,k-1} \\ & + V\left(\sigma^R + \bar{\sigma}^S\right)\bar{N} = V\bar{G} + V\bar{\sigma}^S\bar{N} \end{aligned} \tag{3.110}$$

In the discussion that follows, we include only the i-flows, giving a 1-D result, for the sake of simplification. The terms representing the additional dimensions can be deduced by inspection. Looking more closely at the iteration process, we see that the result of the $p^{th}$ iteration is, where we use the tilde to denote an unrebalanced result that does not satisfy Eq. (3.110):

$$-\left(\tilde{F}^p_{i,i+1} + \tilde{F}^p_{i,i-1}\right) + \left(\tilde{F}^p_{i+1,i} + \tilde{F}^p_{i-1,i}\right) + V\left(\sigma^R + \bar{\sigma}^S\right)\tilde{\bar{N}}^p \tag{3.111}$$

$$V\bar{G} \quad - \quad V\bar{\sigma}^S\,\bar{N}^{p-1}$$

Now, let us assume that each partial flow is proportional to the flux in the cell from which it is emerging:

$$F^p_{i+1,i} = \tilde{F}^p_{i+1,i}\,\bar{N}^p_i/\tilde{\bar{N}}^p_i \quad , \tag{3.112}$$

etc. Given this, a set of fluxes that satisfy Eq. (3.110) can be found from:

$$\begin{aligned} & -\left(\tilde{F}^p_{i,i+1}\,\bar{N}^p_{i+1}/\tilde{\bar{N}}^p_{i+1} + \tilde{F}^p_{i,i-1}\,\bar{N}^p_{i+1}/\tilde{\bar{N}}^p_{i-1}\right) \\ & + \left[\left(\tilde{F}^p_{i+1,i} + \tilde{F}^p_{i-1,i}\right) \Big/ \tilde{\bar{N}}^p_i + V\left(\sigma^R + \bar{\sigma}^S\right)\right]\bar{N}^p_i \\ & = V\bar{G} + V\bar{\sigma}^S\,\bar{N}^p \end{aligned} \tag{3.113}$$

It is helpful to eliminate terms in $G$ using Eq. (3.111). The result can be expressed:

$$\sum_{i'} Z_{i,i'} \bar{N}^p_{i'} = \sum_{i'} Z_{i,i'} \tilde{\bar{N}}^p_{i'} + V\bar{\sigma}^S \left( \tilde{\bar{N}}^p_i - \bar{N}^{p-1}_i \right) \quad , \tag{3.114}$$

where

$$Z_{i,i'} \equiv -\tilde{F}^p_{i,i'} / \tilde{\bar{N}}^p_{i'} \; ; \; i' = i \pm 1 \tag{3.115a}$$

$$Z_{i,i'} \equiv V\sigma^R + \left( \tilde{F}^p_{i+1,i} + \tilde{F}^p_{i-1,i} \right) \Big/ \tilde{\bar{N}}^p_i \; ; \; i' = i \quad , \tag{3.115b}$$

$$Z_{i,i'} \equiv 0 \quad ; \quad i' \neq i \pm 1 \text{ and } i' \neq i \quad . \tag{3.115c}$$

This can be solved by standard matrix-solution routines, producing new fluxes that are in balance with the $p^{th}$ source.

In effect, the residual scattering error in the $p^{th}$ iteration has been redistributed using a transport operator, $Z$, derived from the partial currents. If no further directional redistribution were to take place, the solution would be complete. If directional redistribution is of secondary importance, the solution to Eq. (3.114) serves as a good guess for the next iteration. Expressed in this form, the method can be seen to be an application of the method of synthesis propounded by Kopp[45] and explored by Gelbard and Hageman.[46]

The directional flux can be "fixed up" such as to correspond to $N^p_i$ either by multiplying all directional fluxes at position i by a constant, or by adding a constant to the fluxes. This code 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 $\bar{N}^p$ must be set. Flow from a fixed external boundary source belongs with the $G$ term, and it does not appear in Eq. (3.114). Thus, $F$, at such a boundary, is set to 0. For void, reflected, albedo, and cylindrical boundaries, inward flow at the left boundary is $\alpha$ times the outward flow, where $\alpha$=0 for a void, 1 for reflected or cylindrical conditions, and the

albedo at an albedo boundary. This is implemented by setting $Z_{1,0}$ to 0 and subtracting $\tilde{\alpha} F_{\cdot}^{p} / \tilde{\bar{N}}^{p}$ from $Z_{1,1}$. 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.[47] In the case of a right boundary, inward flow, represented by $\tilde{F}^{p}_{I+1}$, is not proportional to $N^{p}_{I}$ at all. In fact, it is $\tilde{\alpha} F^{p-1}$. A treatment exactly analogous to the left-boundary treatment would set $Z_{I,I+1}$ to 0 and would replace $F^{p}_{I+1,I}$ in Eq. (3.115) with $(1-\alpha)F^{p}_{I+1,I}$. To be entirely consistent, however, a new term must also be added to the right-hand side of Eq. (3.114), with the result:

$$\sum_{i'} Z_{i,i'} \bar{N}^{p}_{i'} = \sum_{i'} Z_{i,i'} \tilde{\bar{N}}^{p}_{i'} + V \bar{\sigma}^{S} \left( \tilde{\bar{N}}^{p}_{i} - \bar{N}^{p-1}_{i} \right) + \alpha \left( \tilde{F}^{p}_{\cdot} - F^{p-1}_{\cdot} \right) \; : \; i = I \tag{3.116}$$

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

An alternate form of Eq. (3.114) is also available. For this, we eliminate terms in $\sigma^{R}$ from Eqs. (3.111) and (3.113), rather than terms in $G$. Defining:

$$f = \bar{N}^{p} / \tilde{\bar{N}}^{p} \quad ,$$

we have:

$$- \left( \tilde{F}^{p}_{i,i+1} f_{i+1} + \tilde{F}^{p}_{i,i-1} f_{i-1} \right) + \left[ \tilde{F}^{p}_{i+1,i} + \tilde{F}^{p}_{i-1,i} + V\bar{G} - V \bar{\sigma}^{S} \left( \tilde{\bar{N}}^{p}_{i} - \bar{N}^{p-1}_{i} \right) \right] f_{i} = V\bar{G} \tag{3.117}$$

In principle, the results of Eq. (3.117) must be identical with those of Eq. (3.114). In practice, the matrix of Eq. (3.117) is easier to solve.

### 3.5.3 Damped Partial Current Rebalance

It is evident, from Eq. (3.117), that negative or divergent $f_i$'s may result if $\bar{\sigma}^S$ is too large. In fact, the application of that method was so troubled with instability that it was customary to perform one or two unrebalanced iterations after each rebalanced iteration.

Based on work by Reed,[48] the problem was 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. In this procedure, the boundary flows are calculated as usual. Then, each is augmented with a fictitious flow as follows, where the prime denotes the augmented flow:

$$\tilde{F}'_{i,i+1} = \tilde{F}_{i,i+1} + F^B_{i+1/2} \quad , \tag{3.118}$$

$$\tilde{F}'_{\cdot\,\cdot} = \tilde{F}_{\cdot\,\cdot} + F^B_{\cdot\,\cdot} \quad . \tag{3.119}$$

and where, in our notation:

$$F^B_{i+1/2} = \frac{z'}{4} \max \left[ V_i \, \bar{\sigma}^S_i \, \tilde{\bar{N}}_i \; , \; V_{i+1} \, \bar{\sigma}^S_{i+1} \, \tilde{\bar{N}}_{i+1} \right] \quad ; \quad z' \geq 1 \quad . \tag{3.120}$$

In a 1978 paper, it was shown that $z' = 2$ gives assurance of non-negative results in 1-D.[49] The same considerations indicate values of 1 and 2/3 for non-negativity in 2-D and 3-D. That paper called Eq. (3.111) the "$\phi$ method," and it also described a "$J$ method," which was a heuristic extension:

$$F^B_{i+1/2} = z' \min \left( \tilde{F}_{i+1,i}, \tilde{F}_{i,i+1} \right) \quad . \tag{3.121}$$

Tests showed this J method to be slightly more effective. With either of these definitions, Eq. (3.117) becomes:

$$- \left( \tilde{F}'_{i,i+1} f_{i+1} + \tilde{F}'_{i,i-1} f_{i-1} \right) \tag{3.122}$$

$$+ \left[ \tilde{F}'_{i,i+1} + \tilde{F}'_{i,i-1} + V\bar{G} - \bar{\sigma}^S \left( \bar{\bar{N}}^p_i - \bar{N}^{p-1}_i \right) \right] f_i = V\bar{G}$$

It was found that values of $z'$, as low as 0.5 give fastest convergence for some 2-D problems, but the optimal value increases with increasing problem difficulty. Better results were obtained in difficult problems by starting $z'$ at 1 and allowing it to increase with an increment of 1 at each iteration. A later study showed that the requirement for a good choice of increment could be mitigated by applying the increment only as needed.[47]

Defining, on a pointwise basis:

$$e = \left( \bar{\bar{N}}^p - \bar{N}^{p-1} \right) / \bar{\bar{N}}^p \quad , \tag{3.123a}$$

$$f = \left( \bar{N}^p - \bar{\bar{N}}^{p-1} \right) / \bar{\bar{N}}^p \quad , \tag{3.123b}$$

and

$\hat{e}$ = value of $e$ where $|e|$ is maximum , (3.123c)

$$\hat{f} = \text{value of } f \text{ where } |f| \text{ is maximum} \quad , \tag{3.123d}$$

then the increment, $\Delta z'$, is applied if and only if

$$\hat{f}\,\hat{e} < 0 \quad \text{(rebalance is opposing iteration)} \quad , \tag{3.124a}$$

$$\hat{f}^p\,\hat{f}^{p-1} < 0 \quad \text{(rebalance is oscillating)} \quad , \tag{3.124b}$$

$$\text{or} \quad |\hat{f}^p| > |\hat{f}^{p-1}| \quad \text{(rebalance is diverging)} \quad . \tag{3.124c}$$

In spite of the arbitrary nature of the damping adjustment, this "partial current rebalance" (PCR) method remains the workhorse of deep-penetration problems, especially at high energy. It is sometimes ineffective at highly scattering epithermal and thermal energies. At worst, the progressive

damping generally converges no worse than the unaccelerated convergence rate, but if $z'$ becomes too large, acceleration will have no further effect.

### 3.5.4 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).[50] 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 in the present application, but they soon led to Aull's "consistent diffusion acceleration" (CDA), which was useful to us.[51] The "mesh-centered" form of CDA can be obtained by simply replacing the $Z$ coefficients of Eq. (3.114) with analogous coefficients calculated from diffusion theory. In slab geometry:

$$C_{i,i'} = -D^T A/\Delta x \quad ; \quad i' = i \pm 1 \quad , \tag{3.125a}$$

$$C_{i,i'} = V\sigma^R - \left(C_{i+1,i} + C_{i-1,i}\right) \quad ; \quad i' = i \quad , \tag{3.125b}$$

$$C_{i,i'} = 0 \quad ; \quad i' \neq i \pm 1 \text{ and } i' \neq i \quad , \tag{3.125c}$$

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 PCR was.

Following Alcouffe, Aull used a "mesh-corner" formulation with good success. Like Alcouffe's method, Aull's formulation was convergent for all cell sizes and all scattering ratios (ratios of scattering cross section to total cross section) less than 1, provided that the diamond difference flux calculation was used and no repair of negatives was employed.[52] In addition, Aull's method was useful in a wide range of practical weighted-difference problems. Although global convergence was not established, the method did not perturb a converged solution, and it was superior to PCR in many cases, especially the calculation of low-energy groups in large, highly scattering regions.[53] Larsen showed, however, that certain extreme conditions could thwart Aull's formulation if a difference method other than diamond is used.[54]

A later study used the mesh-center formulation of Eq. (3.125) with a nearest-neighbor smoothing technique. Boundary conditions are particularly easy to apply in this formulation by adjusting matrix terms, as was done in the PCR formulation. The diffusion coefficient is $P_1$- corrected if $P_1$ data are available, and an upper limit of $3*\Delta x$ is arbitrarily imposed to prevent erratic performance in low-density regions. The correction factors are found for each cell, and then the factors adjacent to each cell vertex are averaged. Finally, the vertex values for each cell are averaged to provide a smoothed

result for the cell. This heuristic method has proven successful in some very difficult problems, occasionally surpassing Aull's formulation.[53] The DORT code has either this method or the comparable mesh-corner formulation available, although only the mesh-corner approach is in standard use.

Certain relationships between CDA and PCR can be seen by comparing Eqs. (3.115) and (3.125). Both methods are equivalent to an equation of the form:

$$T\left(N^p - \tilde{N}p\right) = \sigma^S\left(N^p - \tilde{N}^{p-1}\right) \quad , \tag{3.126}$$

where $T$ is a transport operator based upon the partially converged current-to-flux ratio in one case and upon the cross sections in the other. Now, let us look at the extreme example of the first iteration in a calculation where the source is entirely at the left boundary. In this case, all of the flow will be rightward. As the PCR correction is calculated, it must also flow entirely rightward. That would be a satisfactory prediction of the $p^{th}$ iteration if the material is almost forward-scattering, but it would fail badly if the material scatters isotropically. In fact, some users bypass PCR entirely on the first iteration for some applications. The CDA method, on the other hand, calculates a $P_0$ scattering residual (the right-hand side of 3.126), and then estimates the transport of this residual with a calculation that is $P_1$-correct. That can be a much better solution in highly-scattering problems. Miller came to analogous conclusion in a somewhat different context.[61] It may be interesting to note that additional improvement can be sought by removing the assumption of isotropic residual source. Morel, in effect, allowed the right-hand side of Eq. (3.126) to have a $P_1$, shape, giving superior results in very anisotropic situations.[62]

The damped PCR method was adequate for the problems for which TORT was developed, and it is the only method available for production use at this time. Newer acceleration methods that are compatible with the various flux-calculation methods are being studied, however.

## 3.6 SOURCE ITERATION ACCELERATION

### 3.6.1 Source Iteration Synthesis

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 turn, 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 IIIW code produced by Westinghouse reported a synthesis-style rebalance in the energy variable, but it did not revise the spatial shape. The TWOTRAN-II code solved the alternate problem — an energy-independent space rebalance. A space-energy synthesis seems attractive. TORT presently uses several specialized treatments to be described later, however.

### 3.6.2 Upscatter Rebalance

Upscatter rebalance is the simplest of the techniques. If we sum the terms of Eq. (3.99) over all 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 themselves. Then, with $s$ designating the super group, we have:

$$\overline{\overline{G}}_s \equiv \sum_l \sum_j \sum_k V \sum_{g=g''+1}^{g^{max}} \left\{ \chi_g D + \sum_m W_m \left[ Q_{m,g} + \sum_{g'=1}^{g''} \sum_{m'} W_{m'} \sigma^S_{z,m,g,m',g'} N_{m'g'} \right] \right\} ; \quad (3.127)$$

$$\overline{\overline{H}}_s \equiv \sum_i \sum_j \sum_k V \sum_{g'=g''+1}^{g^{max}} \sum_m W_m \sum_{m'} W_{m'} \sigma^S_{z,m,g,m',g'} N_{m,g'} \quad . \quad (3.128)$$

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

$$f_S = \frac{\overline{\overline{G}}}{\overline{\overline{G}}_S - \left( \overline{\overline{H}}_S^q - \overline{\overline{H}}_S^{q-1} \right)} \quad . \quad (3.129)$$

In practice, this procedure has proven to be often helpful and never harmful. It can not help a problem where the spectrum changes significantly with space, of course.

### 3.6.3 Error-Mode Extrapolation

In an unaccelerated problem, it is typical for a solution to change rapidly at first, as high-order solution modes die away, and then to settle into a pattern of slow convergence, with an almost-constant relative change in fission density between each iteration. Vondy suggested the interpretation of this pattern as the decay of a single error mode, which can often be used to extrapolate toward the correct result.

To accomplish this, the code accumulates the error sum $e$, the eigenvalue estimate, $\lambda$, and an extrapolation parameter, $\Theta$:

$$e^q = \sum \left| D^q - D^{q-1} \right| \ , \tag{3.130}$$

$$\lambda^q = e^q / e^{q-1} \ , \tag{3.131}$$

and

$$\Theta^q = \lambda^q / \left( 1 - \lambda^q \right) \ . \tag{3.132}$$

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

$$D^{q'} = D^q + \Theta^q \left( D^q - D^{q-1} \right) \ . \tag{3.133}$$

The value of $\Theta^q$ is adjusted such that no fission density will have a fractional change greater than an arbitrary limit. Fluxes are also extrapolated in the same way.

Error-mode extrapolation is typically very effective in accelerating $k_{\text{eff}}$ calculations and searches if a stable value of $\Theta^q$ can be found. The method is also helpful in resolving upscatter implicitness, whether the calculation is a fixed-source or $k_{\text{eff}}$ calculation. It is generally compatible with the upscatter rebalance discussed earlier, and the two procedures tend to assist each other. In certain cases, a stable value of $\Theta^q$ can not be established, however, and the method is not helpful in those cases.

### 3.6.4 Subcritical Multiplication Rebalance

The extrapolation may fail in the case of subcritical multiplication problems for which $M$ is very high. Computer roundoff is often to blame. On a machine with 4-byte arithmetic, a problem with $M$>100 may extrapolate at once to a solution estimate very near the correct result, and then not extrapolate again. If the spatial shape of the flux does not change too severely during iteration, a global rebalance comparable to the groupwise and upscatter rebalance can be used. Where $\equiv{Q}$ and $\equiv{D}$ represent total fixed and fission sources, the results of a source iteration are multiplied by the factor:

$$f^s = \frac{\equiv{Q}^q}{\equiv{Q}^q - \left( \equiv{D}^q - \equiv{D}^{q-1} \right)} \ . \tag{3.134}$$

This is generally effective if the source is distributed throughout the fissile material. Extremely localized source distributions can defeat this procedure, however. An example is a system with a thermal neutron source at one edge. The spatial flux shape change can be so severe that global rebalance fails, and it can also thwart the error-mode extrapolation in such a case. For this reason, an option to bypass it is provided.

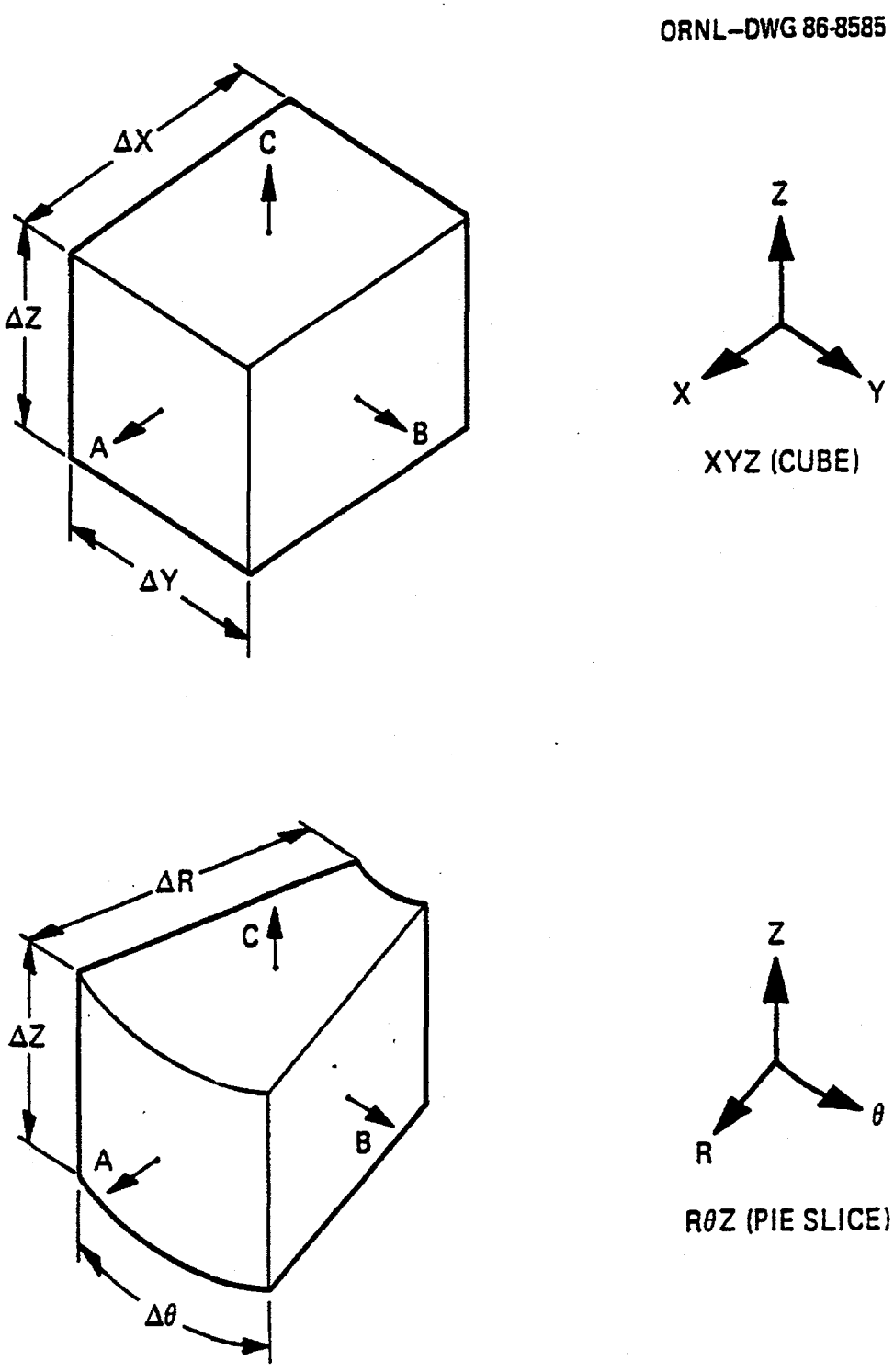



**Fig. 3.1. 3-D Unit Cells.**

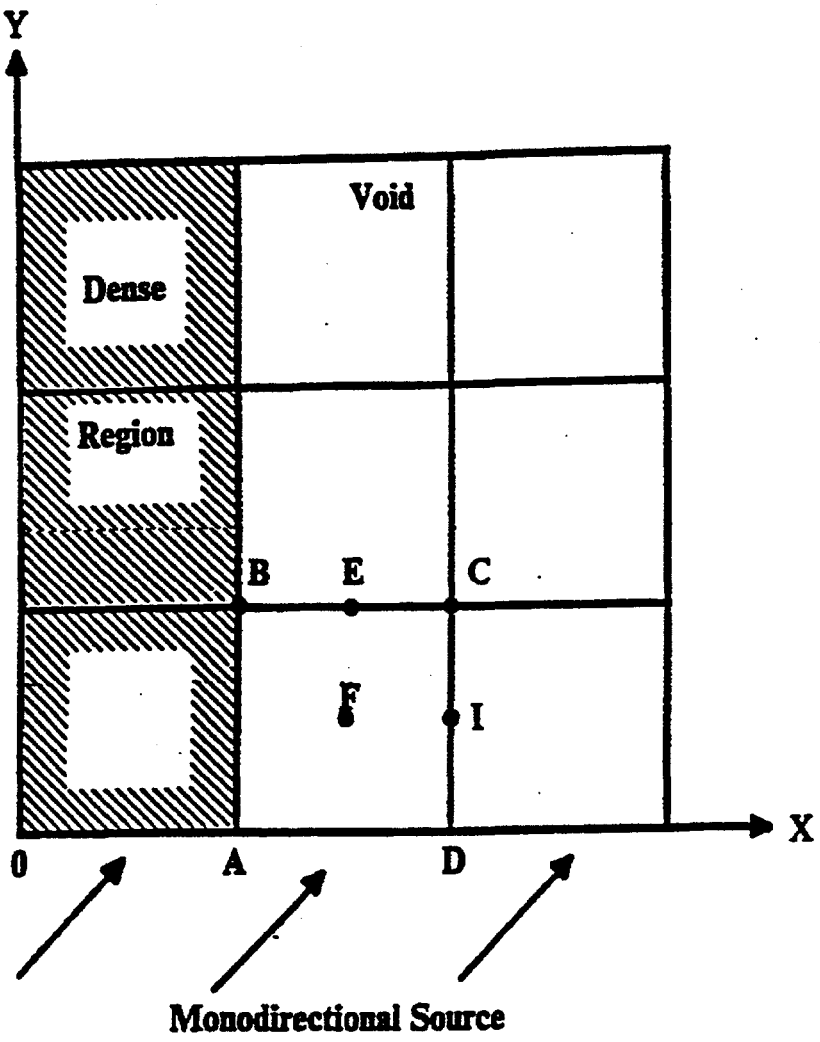



Fig. 3.2. Dispersion problem.

# REGION 4. PROGRAMMER'S INFORMATION

## 4.1 CODE STRUCTURE

The code is arranged in four main structural blocks headed by control subroutines:

1. LOCO - input and edit of parameters controlling problem size and execution path.

2. INPUT - allocation of memory, input and preprocessing of data arrays, error detection, and input file processing.

3. WORK - solution of the transport problem and generation of certain very large output data files.

4. OUTPUT - final normalizing, results summary preparation, writing of all other output data files.

The principal subroutines in these blocks can be seen in the subroutine hierarchy diagram, Appendix H.

The entire list of subroutines used in TORT is given in Appendix H, together with comments as to the purpose of each. The first group, the main structural routines, perform calculations or direct other subroutines in that function.

The second group are special-purpose support routines. In general, these routines perform an extensive, but specialized, job at the direction of only one calling routine. In general, they make full use of the common blocks as well as information contained in argument lists. A few are shared with the 2-D code DORT, on which TORT is based, as indicated in the appendix.

Next, general-purpose support routines are listed. They perform very limited tasks based upon data supplied in the argument list. They tend to be called from many places in the code. All are shared with DORT.

Following these routines, system-dependent interface packages are listed. These packages contain calls to library routines specific to a particular computer and special machine-dependent coding required for acceptable performance on a given machine. The Environmental Information region discusses these packages further.

Any of the system-dependent packages may need local modification on systems differing from those on which they have been tested. They contain extensive internal documentation to make this job feasible. In fact, it has not been difficult for users to modify an existing package for an entirely new computer type.

Overlay arrangements are not provided with the present code, and they are not necessary or desirable for the systems on which it is presently used. It would appear straightforward to construct an overlay structure from the information in Appendix H, however.

## 4.2 SPECIAL DIMENSIONING PRACTICES

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 and LCHI is the array locator corresponding to array CHI, CHI is made available to a subroutine by, for example:

```
LCHI = ...
CALL SUB( D(LCHI), N, ...)
        .
        .
        .
SUBROUTINE SUB( CHI, N, ...)
DIMENSION CHI(*), ...
X=CHI(N)
        .
        .
```

This is in keeping with language standards, and it operates efficiently on every known computer. Much the same goal can be accomplished with pointers, but pointers have not provided the same efficiency in tests conducted to date.

Many routines use the "pivot array" method rather than multiple dimensions. While this practice is also within the limits of the language standards, it deserves clarification. For example, suppose IM(J) is the limit of I for each J, JM is the limit of J, and:

IBJ(J) = sum of IM(JJ-1) for JJ=1,...,J ,

IM(0) = 0 ,

and IMSJM = IBJ(JM+1)) .

Then, the value of an array IJZN corresponding to I and J is found as follows:

IZ = IJZN(I+IBJ(J)).

This can be extended to more dimensions, of course. It is the basis of the "discontinuous space mesh" feature described in Appendix D.

## 4.3 OTHER SPECIAL PRACTICES

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 temporary parameters whose use is entirely local from parameters of real significance, to which mnemonic names are attached. They are also used where a few contiguous variables are needed. It is possible that this ancient practice might inhibit optimization on some modern compiler, but examples of that have not been demonstrated.

## 4.4 MACHINE-SPECIFIC CODING

In some cases, separate statements are required on different computers in order to provide safe and efficient operation. One of the best examples is the special need of Cray computers for vector instructions. Other examples include special treatment of roundoff difficulties and adaptation to different machine word lengths. Where parallel coding is required, the sections are identified and controlled by conditional compilation directives imbedded in the code. The CMP pre-processor selects the correct section of coding for the chosen machine, and it does not copy sections of coding which are inappropriate.

Additional information as to the application of CMP is given in the Environmental Information region and in the comment files at the beginning of the source material.

## 4.5 PORTABILITY CONSIDERATIONS

The portability goal is to make the code as portable and as adaptable as possible without important sacrifices in capability. The spirit of the ANS-STD.3-1971[63] guidelines have been helpful in this. These guidelines advise adherence to a language standard except where deviations provide essential improvement in capability, can be localized, and can be documented. The FORTRAN 66[64] standard was in effect at the time of that document, but the concepts apply equally well to newer standards.

Interface files that may transfer data into or from the code are constructed and documented in the spirit of the CCCC working group recommendations[65,66], and the I/O and timing routines follow the guidelines originally laid out by the group, insofar as practical.

## 4.6 LANGUAGE STANDARDS

The language objective is to use FORTRAN 77[67] insofar as practical. Exceptions to that rule are:

- A few FORTRAN 90 extensions that do not cause difficulty on existing compilers are used, and they are discussed in the comments to follow.

- Some portions of the code were created in FORTRAN IV, long before 1990 or even 1977, and they have not been completely upgraded.

- In some cases, the nature of external file formats force non-standard features.

- Limited use of C language is made in conjunction with the use of system library subroutines.

- Limited use of assembly language is made to speed execution on Cray computers. The code can operate without these routines, however.

FORTRAN 90 extensions include:

- compatibility with lower-case characters (the code may be compiled in either upper-case or lower-case characters if input data are adjusted accordingly),

- variable names longer than 6 characters, although a limit of 8 is generally observed,

- the INCLUDE statement, and

- the block DO statement using ENDDO.

Certain practices have been carried over from FORTRAN IV. While these do not cause difficulty on systems on which the code is presently run, it may be important to list and understand them. It is the intention, of course, to upgrade them as soon as practical. They include:

- Modern standards provide that unessential dimensions, i.e., the length of singly dimensioned arrays and the last dimension of multi-dimensioned arrays, can be set to "*". In older routines, these may be set to "1" instead.

- Frequently, the name of the container array appears in a routine together with another array that is a subset of the container. Since this is necessarily widespread, no special comment is given of this fact. A comment is given at the beginning of a routine in which any other dual reference to the same storage location occurs, however.

- Real and integer variables must have the same word length. In a few instances, real statements such as A(I) = B(I) may be used to move integer data from one location to another without change, but A(I) = -B(I) or A(I) = 0 would not be allowed. The potential difference between real and integer 0 is recognized. I/O routines may use a single variable type to move either real or integer data between memory and external files.

- Hollerith constants are used in older sections of the code, and the data statements defining them may use either the '...' or the nH... type of specification.

- The FIDO input routines expect that a character variable with length equal to the number of bytes in a real or integer variable maps exactly to a real or integer such that a character argument in a calling routine can appear to be a real or integer variable in the called routine.

## 4.7 MEMORY AND EXTERNAL FILE ADAPTATION

### 4.7.1 Flux/Source Storage Adaptation

TORT is intended to operate with good efficiency in several dissimilar environments, and this requires adaptability in the storage and management of its data. Much of the data storage is devoted to flux- and source-moment arrays. Accordingly, several modes of storing these arrays are provided. The simplest is the problem-stored mode (NJBLK = 0). All flux and source moments are stored in the working memory and are addressed in place. This mode minimizes I/O costs, but the memory requirement may exceed the maximum specified by the user as LOCOBJ. For example, the space required for a 100,000 mesh cell problem with 58 energy groups is roughly 26 megawords -- well beyond the memory of many systems. On the other hand, problems using 3 million mesh cells are not uncommon now.

If the problem can not be stored in LOCOBJ words using the problem-stored mode, the group-stored mode (NJBLK = 1) is tried. An external file is created for the entire volume of flux moment data, and a "user buffer" is allocated at the end of the problem memory to contain the data for one energy group at a time. Two additional buffers of comparable size are required for source data, although these buffers are not associated with an external file in this mode.

Using this mode, considerable I/O work is required to calculate the source scattering into a group from other groups, but the flux iterations for that group can proceed uninterrupted by I/O of source and flux moments. The memory requirement for the 100,000-cell problem mentioned above would be 1.8 megawords in this mode -- manageable on both mainframes and workstations. This mode often provides an optimal combination of memory usage and I/O activity, if enough memory is available.

Should the "group-stored" mode prove to be inadequate based on the storage available, an external file is allocated to hold the coarse mesh rebalance coefficients and the core memory requirements reduced correspondingly. The requirements for the "group-stored" mode are then redetermined based on this reduced memory requirement.

As problem size increases, other measures become necessary, and the plane-stored mode (NJBLK > 1) is invoked. Two additional external files are allocated to hold the source moments for one group, and the user buffers are reduced in size by breaking the space mesh into NJBLK blocks of JBLK1 planes each. The blocks are moved into and out of memory as required to follow the sweep of the planes from top to bottom and back to top for each flux iteration.

A table of flux/moment storage requirements will serve to quantify the effect of blocking:

| MODE | WORKING MEMORY | USER BUFFERS | EXTERNAL FILES |
|---|---|---|---|
| Problem stored (NJBLK = 0) | IM*LM*JM*KM*(2+IGM) | 0 | 0 |
| Group stored (NJBLK = 1) | IM*LM*3 | IM*LM*JM*3*KM | IM*LM*JM*KM*IGM |
| Block stored (NJBLK > 1) | IM*LM*3 | IM*LM*JM*3*JBLKI | IM*LM*JM*KM*(2+IGM) |

The zone map information is kept in memory for NJBLK = 0 or 1, but on an external file otherwise. The input/output of boundary sources and fluxes, distributed source, and past scalar flux information is performed between planes in all modes.

The input parameters MINBLK and MAXBLK can be used to modify this blocking. If MINBLK > 0, blocking will continue until NJBLK is at least as large as MINBLK. If MAXBLK is entered, and if NJBLK exceeds MAXBLK, then the problem will be terminated with the message, "Insufficient space for this problem."

If NJBLK reaches KM, the space requirements exceed LOCOBJ, and MAXBLK has not been exceeded, then a message will be given, "LOCOBJ too small, continuing." At that point, unless the user has set NCNDIN = 0, the system will attempt to increase the memory partition to the actual requirement. This seldom succeeds on UNIX-based systems, however, and an error message will generally be given from DLOCAL or MEMORX that the requested memory is not available.

If LOCOBJ is too small to contain all of the input data arrays, the message, "LOCOBJ too small, terminating," will be given, and the problem will stop at once. This rarely happens, however.

If the problem requires less than the full machine memory, some tradeoff is available in the use of memory vs. I/O. Specifying the largest memory size available as LOCOBJ will result in the minimum I/O and the shortest CPU time. This may reduce the usefulness of the machine to other users, however. It may also incur larger charges when they are based, in part, on memory. A smaller memory partition might be preferable in these cases.

If a large value of LOCOBJ is specified, and if less memory is needed, the code will request the return of the unused memory to the system. On some systems, this request will be ignored, however.

### 4.7.2 Memory Requirements

The full list of data arrays to be stored in memory is lengthy, but a few larger arrays dominate the total. The tables below allow a quick estimate of the requirement. Exact information is available by studying subroutine INPUT.

The memory is allocated in blocks which may overlay each other during different phases of execution:

| Block | Use | Phases Needed |
|---|---|---|
| L0 | Mesh description | All |
| L1 | Cross sections, zone map | All |
| L2 | Fission, flux and source for a row, internal boundary source | All |
| L3 | Acceleration data | Flux sweep, acceleration |
| L4 | Boundary fluxes and shapes, source moments, scratch for flux sweep | Flux sweep |
| L5 | Scratch for acceleration | Acceleration |
| L6 | Source input and edit | Input |
| L7 | Boundary source input | Input |
| L8 | Response summary | Output |
| L9 | User buffers | After source input and before output |

The important arrays contained in the blocks are listed in Table 4.1. The arrangement of the blocks is illustrated in Fig. 4.1. LSTR0 is the end of space taken by input arrays and certain data derived from them. LENDR must be positioned to allow space for LEND2, LEND3, and LEND4, all followed by L9. Some of these key endings can be identified in the printout:

| Symbol | Printout Identification |
|---|---|
| LEND1 | END OF GEOM ARRAYS |
| LEND2 | END OF SWEEP ARRAYS |
| LEND3 | END OF REBAL ARRAYS |
| LEND4 | END OF SOURCE ARRAYS |
| LENDR | END OF WORK ARRAYS |
| LEND11 | END OF SOURCE I/O ARRAYS |
| LEND12 | END OF OUTPUT ARRAYS |
| LENDF | END OF USER ARRAYS |
| LENDB | END OF OUTPUT ARRAYS |

## Table 4.1. Principal Data Arrays in Memory

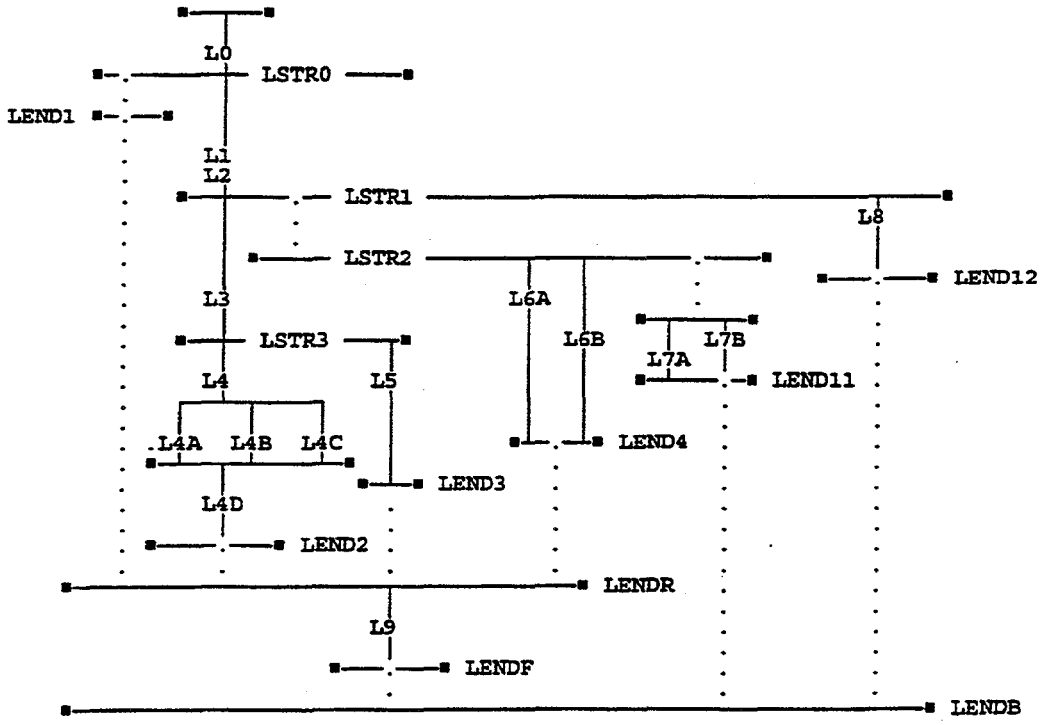
| | | | | |
|---|---|---|---|---|
| L0 | ISET | JM*KM | i-mesh set # by (j'k) | |
| L0 | IMBIS | JM*KM | # of cells in each i-mesh set | IM.lt.0 |
| L0 | IBJK | JMSKM | pivots for (i'j'k) indexing | |
| L1 | SIG | IHP*NCRX*(NSCTM+l) | cross sections | |
| L1 | IJZN | IMSJM*KM | zone by mesh cell | NJBLK.le.1 |
| | | IMJSJM | | NJBLK.ge.2 |
| L2 | FSNIJ | IMJMKM*(1+3*NODAL) | new fission density | CHI array ne.0 |
| L2 | FSOIJ | IMJMKM | old fission density | CHI array ne.0 |
| L2 | FLUM | IMA*LM*JM*KM | flux moments | NJBLK.eq.0 |
| | | IMA*LM | " | NJBLK.ge.1 |
| L2 | SINM | same as FLUM | inscatter source moments | same as FLUM |
| L2 | SORKT | MMDNUP*IMA*JM*NKNTSR | internal k-boundary source | |
| L3 | CPCX | ICM*JCM*KCM | rebalance data | |
| L3 | CPCL | ICM*JCM*KCM | " | |
| L3 | CPCR | ICM*JCM*KCM | " | |
| L3 | CPCI | ICM*JCM*KCM | " | |
| L3 | CPCO | ICM*JCM*KCM | " | |
| L3 | CPCB | ICM*JCM*KCM | " | |
| L3 | CPCT | ICM*JCM*KCM | " | |
| L3 | CPCOX | 2*JCM*KCM | " | |
| L3 | CPCOY | 2*ICM*KCM | " | |
| L3 | CPCOZ | 2*ICM*JCM | " | |
| L4 | FIO | MMA*JM | boundary ftux | |
| L4 | FJO | MMA*IMA | " | |
| L4 | FKO | MMDNUP*IMA*JM | " | |
| L4 | CIY | MMA*JM*NODAL | nodal boundary shape | |
| L4 | CIZ | MMA*JM*NODAL | " | |
| L4 | CIX | MMA*IMA*NODAL | " | |
| L4 | CIZ | MMA*IMA*NODAL | " | |
| L4 | CKX | MMDNUP*IMA*JM*NODAL | " | |
| L4 | CKY | MMDNUP*IMA*JM*NODAL | " | |
| L4 | SORM | same as FLUM | total source moments | same as FLUM |
| L4 | FII | MMA*JM*IFBC | scratch for PLANE, SBSFIX | |
| L4 | FJI | MMA*IMA*IFBC | " | |
| L4 | FIP | IMSJM | " | |
| L4a | FMO | MMDNUP*IMA*JM | scratch for remeshing, SORX | IM.lt.0 |
| L4b | FMO | MMA*(5+IMA) | scratch for remeshing, SORX | NEGFIX.gt.0 |
| L4c | FMO | IMA*LVLMX | scratch for ROW | |
| L4d | FIX | IMA*(22+70*NODAL) | scratch for ROW | |
| L4d | DIRF | IMA*MMDNUP/2 | directional flux/source | NEGFIX.gt.0 or NTDIR.gt.0 |
| L5 | CPCF | ICM*JCM*KCM | scratch for CMSCLR | |
| L5 | CFAC | ICM*JCM*KCM | " | |
| L5 | SBAN | ICM*JCM*KCM | " | |
| L5 | EPSCM | ICM*JCM*KCM | " | IEPSBZ.gt.0 or IACCl.eq.1 |

| | | | | |
|---|---|---|---|---|
| L6a | FLIJ | IMJMKM | source input & edit | |
| | | IMJMKM | " | mod(NEDSOR/100,10).gt.0 |
| | | (NREG+1)*(IGM+1) | " | mod(NEDSOR/10, 10).gt.0 |
| L6b | FLIJ | IMSJM | source input | |
| L6b | FBS | MMA*IMA*JM | " | |
| L6b | FBI | MMA*(IMA+JM) | " | |
| L7A | SII | MM*JM*KM*IFBC | boundary source input | |
| L7A | sil | MM*IMA*KM*IFBC | " | |
| L7A | FIN | MM*JM | " | NBCL=4 or NBCR=4 |
| L7A | FJN | MM*IMA | " | NBCI=4 or NBCO=4 |
| L7B | SKI | MMDNUP*IMA*JM*IFBC | " | |
| L8 | FRES | IMJMKM | response | NRESP.ne.0 |
| L8 | CLRES | IMJMKM | " | " |
| L8 | FKRES | IABS(NKEYFX)*NRES | " | " |
| L8 | ZNRES | NREG*NRES | " | " |
| L9 | LBUF | IMA*LM*JM*JBLK1*3 | flux/source user buffers | NJBLK.ge.1 |

Notes:

| | | |
|---|---|---|
| IACC1 | = | mod(IACC/10,10) |
| ICM*JCM*KCM | = | number of coarse mesh ceLLs |
| IFBC | = | 1 if IBL, IBR, IBI, 1BO, 1BB, or IBT=4; else 0 |
| IHP | = | IHM+1 if IUPS>0; else IHM |
| IMA | = | iabs(IM) |
| IMSJM | = | IMA*JM |
| IMJMKM | = | # of cells in the space mesh |
| JMSKM | = | # of rows in the space mesh |
| LVLMX | = | the number of eta levels |
| MMA | = | iabs(MM) |
| MMDNUP | = | the maximum number of directions downward or upward |
| NREG | = | # of edit regions |
| NRES | = | iabs(NRESP) |
| NODAL | = | 1 if the nodal or characteristic method is used; else 0 |

Other variables are explained in later sections.




Note: Vertical dotted lines indicate no arrays or arrays of trivial length.

Figure 4.1. Memory Block Arrangement.

# REGION 5. USER'S INFORMATION

## 5.1 INPUT DATA SPECIFICATIONS

### 5.1.1 Data Record Input Format

The formatted input data for a problem consist of a title record followed by 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 a symbol indicating real (**) or integer type ($$).

A separator record must be placed between successive problems. If the separator contains the word "dump" as the first four characters, a traceback will be given by subroutine ERRO on machines that offer that service, followed by a stop with status code 777. If the word "=end", "stop", or "end " appears, execution will stop with a status code of 7. Otherwise, input for the next problem begins immediately. No data are retained in memory between problems. At the end of the data stream, execution normally terminates with a stop and a status code of 0. Any system message indicating that the end-of-file has been reached indicates an abnormal condition.

All of the formatted input except the title and separator records are read by the Floating Index Data Operation (FIDO) input processor. The format required for FIDO data is described in Appendix A. The data arrays and blocks to be read are described in following sections of this region. Many arrays are to be supplied only under special circumstances, such as the use of particular options. Unneeded arrays may always be omitted, in which case default values, if any are noted, will be used.

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 are of improper length or if the data are otherwise unsuitable. ERRO prints an explanation of the error and sets an error flag. Data are edited as soon as practical after each block is completed.

Processing of input continues for as long as possible, even after a relatively minor error. In most cases, this allows the user to test all of the data blocks in spite of early errors. Execution is terminated at the end of input processing if a sufficiently severe error has been encountered. It is possible that an early error will cause subsequent data to be erroneously flagged as wrong. An error can also cause subsequent system-related failures, such as addressing or storage errors. Thus, the user must consider whether a given error or failure could have been produced by a previous error.

### 5.1.2 Input Array Specification

#### 5.1.2.1 Overview of Array Requirements

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

1. Control parameter input arrays (61-70) that set array lengths explicitly and control execution options.

2. Primary input arrays (71-80) whose lengths depend upon 61-70, and whose contents determine later array lengths.

3. Secondary input arrays (81-90), whose lengths depend upon previous arrays and whose contents determine later array lengths. (These include the directional quadrature set specification, coarse mesh specification, and assignment of material zones to edit regions.)

4. General input arrays (1-60), which do not determine the lengths of other input arrays. (These include space-mesh specifications, zone descriptions, and much more.)

These four blocks are always required, although unneeded arrays may be omitted. In fact, a block may contain no arrays at all, just its "t" delimiter. Blocks after the first four, containing arrays numbered 91 and higher, allow simple sources to be described without the need for special input files. All of these blocks are optional, depending upon input control values.

The following portions of Subsection 5.1.2 detail the specific array requirements and the conditions under which optional arrays and blocks are to be entered. A brief discussion of certain items is given in the remaining portions of Section 5.1. Keywords appearing in italics are explained in more detail in later sections of Region 5.

#### 5.1.2.2 Detailed Array Specifications

The detailed array specifications are given in Table 5.1 (pg. 5-31).

### 5.1.3 Input/Output File Logical Unit Numbers

The input/output file control array (61$$) controls the designation and use of files on external devices. The flux moment file contains flux moments for each space cell and discrete directional flux values at the external boundaries. This file is sufficient for an exact restart of a previous problem, and it is valuable for use in post-calculation analysis or display codes as well. Specifying a unit number as NTFOG causes the file to be written. Specifying NTFLX causes a previously-written file to be used as an input flux guess.

The *cross section input* unit (NTSIG), with a default value of 8, is always required.

If NTBSI and/or NTDSI are entered as positive numbers, the corresponding files must contain *fixed-source input* data supplied by the user. If NTBSI is negative, a source is to be synthesized on the internal scratch files based on the 90+ arrays. If NTDSI is negative, a temporary scratch unit must be supplied for a source generated by the code based on the 90+ arrays.

If NTRSO, is entered, pointwise *response summaries* will be written on that file. A negative number will cause the output to be written as formatted data on that unit.

If NTNPR is entered, large array edits will be shifted from the standard output to that unit.

If NTSCL is entered, a scalar flux output file will be written on that unit. This file is intended mainly for post-processing codes, and it contains control and parameters and space-mesh information to expedite that use. It can also be used as a restart, but, since it does not contain flux moments or boundary fluxes, most problems will suffer from the absence of that information.

If NTZON is entered, the zone map is to be read from that file, rather than entered from the 8$$ and 14**-19** arrays.

If NTDIR is entered, *directional flux output* will be written on that unit for a "patch" of the space mesh designated by the 34** array.

### 5.1.4 Procedure Control Integers

The control integer array (62$$) begins with two parameters that control limits in the *flux iteration convergence* procedure. NSRMX is the limit on source iterations, normally 1 on fixed-source problems (KTYPE = 0) unless upscatter or fission are present. NFXMX controls the number of flux iterations performed on each group at each source iteration. If NFXMX < 0, the flux iteration limit is taken from the 21$$ array. IEPSBZ specifies the use of the zone importance array (20**) in convergence tests and edits.

ICSPRT and IFXPRT control large cross section and flux array printouts. If MODEP is set to 1, the flux iteration monitor line will be omitted. This is useful for K-calculations. NKEYFX specifies the number of *key flux positions* used to monitor convergence, while NRESP is the number of *response summaries*.

KTYPE, NSCTT, IADJ, and NEGFIX control the flux solution procedure. If KTYPE = 0 and NSRMX = 1, a basic calculation of flux resulting from *fixed-source input* is performed. If fission cross sections are present and a non-zero flux guess is supplied, the fission will be included with the fixed source in determining the flux. If fission is present and NSRMX > 0, a *subcritical multiplication* calculation is done. The code iterates on successive approximations to the

equilibrium flux resulting from the fixed source. *Upscatter* also requires NSRMX > 0 for its convergence. When KTYPE = 1, $k_{eff}$ is calculated, and NSRMX > 0 is generally required for that case, also. An initial flux guess or distributed source results in an initial fission density, and the density is then iterated to convergence. Values of KTYPE > 1 and NSRMX > 1 indicate *indirect searches*, in which the system configuration is altered in order to obtain a value of $k_{eff}$ specified by EVOBJ.

If the default value of NSCTT is used, flux moments corresponding to all components of the anisotropic *cross section input* expansion are calculated. NSCTT can specify a smaller order of expansion to be used in the actual calculation, however. This may save CPU time, make the moment file smaller, and/or allow the user to study expansion effects. IADJ specifies the adjoint calculation procedure. This requires understanding of the unusual *adjoint ordering* of the 90+ arrays and of the output. NEGFIX allows *negative source removal* to be applied, usually improving convergence, and often improving accuracy.

NSCLMX and NBFACC are controls used in the *flux iteration acceleration* procedure. NSCLMX limits the number of acceleration iterations performed after a flux calculation, while NBFACC specifies the number of initial flux iterations performed without acceleration. Setting NBFACC = 1 can avoid certain acceleration problems that may occur due to the poor partial-current information available in the first flux iteration. This can be especially significant if a problem contains large internal voids. NBFACC is ignored after the first source iteration.

MODE selects between different flux-sweeping formulations. In general, standard and alternate procedures should give essentially equivalent results, but standard is expected to execute fastest on the machine being used.

LOCOBJ, LCMOBJ, MINBLK, and MAXBLK control storage adaptation for a particular environment. NCNDIN sets the maximum *user error code* under which a problem is allowed to continue execution. INGRPS controls the number of flux guess groups to be read from NTFLX. This can be used to restart a problem from an incomplete flux file. INMOMS allows the length of the *flux moment expansion* to be altered in a restart; e.g., a P-1 problem could be restarted in P-3 by specifying the proper number of moments here. This feature also allows a weighted-difference problem to be restarted in nodal mode or vise versa. INDIRS allows the boundary fluxes to be ignored on restart so that a different direction set can be used.

NKNTSR specifies the number of internal k-boundary sources to be used. IACC selects the mode of *flux iteration acceleration.*

ITALLY can produce a helpful summary of CPU and problem charge time usage at the end of each problem. It may increase the cost of the problem significantly, however. NCPU controls multitasking on Cray platforms running the UNICOS operating system: NCPU = 0 avoids parallelization altogether; $1 \leq$ NCPU $\leq 16$ executes the Direction Parallel algorithm on the master

and (NCPU -1) slave tasks. NEDSOR allows *fixed source input* to be entered as a 3-dimensional distributed-source array, or as a product of three space vectors, in the 90+ arrays. It also allows two choices of edits for the distributed source. NIFCNV is used with the *accurate restart procedure.* NIFBND allows the boundary source to be supplied in a pre-written file as a part of the *direct access scratch file retention* feature.

### 5.1.5 Geometry Control Integers

IM, JM, KM, IZM, and MMESH control the specification of *space mesh* data. If IM < 0, a *discontinuous mesh* is indicated. IBL, IBR, IBI, IBO, IBB, and IBT control the *boundary conditions* to be applied at the system boundaries. INGEOM controls the geometry choice. The two-dimensional geometries are used primarily for comparison with the DORT code. Comparison of XZ and XY results is also helpful in ensuring symmetry. MM is the length of the *directional quadrature set.* If MM < 0, a *variable quadrature* is to be specified by energy group. In that case, a *variable cross section expansion* can also be used.

NISM and NJSM are special options used with the *discontinuous mesh* capability in order to specify special mesh sets for the boundary source input and storage.

### 5.1.6 Cross-Section Control Integers

IGM, NCRX, NSCTM, IHT, IHM, IUPS, and NEUT describe the *cross-section input*, and are determined from the nature of the cross-section library supplied as NTSIG.

### 5.1.7 Procedure Control Reals

EPP, EPF, EPO, and EPK are the *flux iteration convergence* and *source iteration convergence* parameters. CONSCL, CONACC, CSOLCN, CSOLIN, and EPSCL are controls for the *flux iteration acceleration* and *flux iteration stabilization* procedures. SORNEG establishes a lower limit for the *negative source removal.* FLXMIN defines a level of triviality below which fluxes are deemed too small for application of the convergence tests. THETA, WMAX, and WNODAL are controls for the *flux sweep formulations*. EVI, EVDELK, DEVDKI, EVCHM, EVMAX, EVKMX, and EVTH are controls for the *indirect searches* (KTYPE ≥ 2). EPXTR is used in the *source iteration acceleration.*

### 5.1.8 Problem Control Reals

XNF is the multiplier used in *source normalization.* Its use depends upon the value of KTYPE. TMAX is the job *time limit* in minutes. EKOBJ is the value of $k_{eff}$ sought in indirect searches (KTYPE ≥ 2). For $k_{eff}$ searches (KTYPE = 1), the value affects one of the output edit columns, but nothing else. It has no meaning when KTYPE = 0. The value of EKI establishes an initial $k_{eff}$ estimate, largely useful when restarting a $k_{eff}$ calculation which has been partially converged

previously. In such a case, EKI is used to normalize the input flux guess, making the first source iteration more efficient.

### 5.1.9 Primary Dimension-Setting Arrays

Arrays in this block are used in the *discontinuous mesh, variable quadrature, and variable cross section expansion* features.

### 5.1.10 Secondary Dimension-Setting Arrays

This is the final block for arrays on which the length of other arrays depend. The 81**-83** arrays are the *directional quadrature set.* The 84**-86** arrays determine the coarse *space mesh.* A good choice of coarse mesh can reduce memory requirements and costs in certain problems. IZNRG assigns region designations to material zones.

### 5.1.11 Problem Specification Arrays

In this block, all remaining arrays except optional source-generation arrays are located. The 1** array is the fission spectrum, while 6** is the spectrum used in the *fixed-source input.* The 2**-4** are *space-mesh* boundaries, while 5** is the set of energy-group boundaries, presently used only in certain output edit files. The 7**, 8$$, 9$$. and 14**-19** arrays relate the cross sections to the space mesh through the *zone and material specification by body overlay.* The 20** array allows the specification of *flux iteration convergence* and *source iteration convergence* importance by zone. The 21$$ array allows the flux iteration limit to vary by energy group.

The 22$$-24$$ arrays specify a set of *key flux positions* for which convergence can be monitored at each iteration and printed after the last. *Response summaries* can also be obtained at these locations. The 26**-27** arrays define zone- and group-dependent responses at those points, as well as in other edit options.

The 30$$ array specifies the location of the internal boundary *fixed source input.* The 34** array specifies the bounds of a "patch" of the space mesh selected for *directional flux output.*

### 5.1.12 Source Specification Arrays

The remaining blocks, those containing the 90+ arrays, are optional, providing a means of generating source files in certain simple cases. Cases not described by these blocks can be treated by supplying sources on input files generated by the user.

## 5.2 INPUT DATA DISCUSSION

### 5.2.1 Space Meshes

Five different subdivisions of the problem space are determined from the input data. Three sets of fine-mesh interval boundaries (2**, 3**, and 4** arrays) are input, corresponding to the three spatial dimensions. The intersections of these boundaries form the fine-mesh cells. (In curved geometry, the $\theta$ is entered in units of revolutions, not radians or degrees.)

A coarse mesh, specified by three additional sets of intersecting boundaries (84**, 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. Each coarse-mesh boundary must also be a fine-mesh boundary.

Material zones 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 assignments, certain input and edit features depend upon material zones. Material zones are specified, and material numbers are assigned to them, by the use of body overlay, to be described later.

Edit regions (87$$ array) are groupings of material zones used for the purpose of condensing certain output tables.

### 5.2.2 Cross Section Input

The cross sections supplied on unit NTSIG are to be supplied according to the GIP format (Appendix B). This format specifies ordering by table position, then by component, then by material number, and then by group. MTM specifies the number of materials required. Each material describes a specific isotope, isotopic mixture, or a mixture of elements. The information must be in macroscopic, not microscopic, form. Within each material, NSCTM+1 components are required. The first component describes the total probability for a given process. Subsequent components describe the angular distribution of the particles emerging from that process, expanded in Legendre basis functions. In general, more components result in more accuracy, at the cost of more CPU time and larger moment files. If a lower order of expansion in the calculation is desired, this can be achieved without changing the cross section input, by entering a non-negative value of NSCTT. Within each component, IHP table entries are required, each describing a specific process. If IUPS=0, IHP=IHM. Otherwise, IHP=IHM+1 to allow inclusion of the total upscatter from a group as entry IHP. Cross section preprocessors such as GIP perform this task, and it requires no special action on the part of the user. The ordering of the IHP entries for a non-adjoint problem is listed in the appendix describing file formats, and the modifications necessary for an adjoint problem are specified there. The documentation for the GIP cross section processor[6] also illustrates the ordering of data for an adjoint calculation.

### 5.2.3 Flux Moment Expansion

The use of cross-section components beyond the first requires the use of a flux moment expansion describing the directional distribution of the flux in each space cell. The first term of the expansion is the conventional scalar flux, the next three are directional currents, and subsequent terms allow more-complicated shapes to be described. When weighted difference is used, the length of the expansion, LM, is:

$$LM = L*L ,$$

where L is equal to NSCTT+1 if NSCTT $\geq$ 0, and NSCTM+1 otherwise.

If the nodal formulation is used, additional terms are required to describe the spatial variation of flux within the cell, and LM is increased to:

$$LM = L*L+3 .$$

If a problem is restarted with a flux guess from a problem for which the value of LM was different, the old value of LM must be entered as INMOMS. Moments beyond INMOMS on the input file are ignored. Moments required by the problem but not found on the input file are initialized as 0.

### 5.2.4 Negative Source Removal

Any problem using L > 1 may incur negative source generation. This problem is simply due to the failure of a truncated Legendre expansion to describe adequately the angular shape of the cross sections. If the problem has no areas of severe anisotropic flow, the truncation effect may never be noticed. In certain difficult problems, however, the negative sources can result in severe errors or even negative scalar flux. When NEGFIX $\neq$ 0, however, negative sources are removed immediately when they are generated.

The "economy fixup" procedure (NEGFIX = -1) sets the negatives to 0 without attempting to maintain particle conservation. It adds almost nothing to the CPU cost of a problem. It is especially valid in deep-penetration calculations where an over-estimate of flux is acceptable. It does violate conservation of particles, however, and it could cause problems in a $k_{eff}$ search.

An alternative "conservative fixup" is used in the other options, but at potentially significant CPU cost. The "full fixup" option (NEGFIX = 1) searches for negatives during the first iterations of each energy group and repairs them by deleting both negatives and small positive sources in such a manner as to conserve particles. In many cases, the small positives deleted are the result of the same oscillations that produced the negatives, and their removal does no harm. This conservative method, although heuristic in nature, has proven accurate in some severe tests. It may add 20% to the CPU cost of a problem if few negatives are found, however, and much more if the negative contamination is high.

To minimize the costs, the search is abandoned for a group after a given iteration has yielded no negatives. Experience has shown that negatives that disappear during a given iteration do not generally reappear in subsequent iterations. The "economy fixup" procedures still remain in effect, in any case, as a final line of protection.

The "initial fixup" (NEGFIX = 2) operates like the full fixup, except that, after no negatives are found in any iteration of a given energy group, the search is abandoned for all subsequent groups of that particle type. Experience has shown that, after a group has been solved without negative contamination, lower groups will generally be uncontaminated. It is important, with this option, to set the value of NEUT properly, so that the search will begin again with the second particle type, if any.

If NEGFIX = 3, the action is as described above, except that a fractional negative contamination less than SORNEG is ignored. This would be used to avoid the cost of removing trivial negative contamination.

### 5.2.5 Zone And Material Specification by Body Overlay

Three-dimensional problems are likely to involve large meshes and intricate zone shapes, so that special techniques are required in the input. In TORT, the 9$$ array assigns a specific material to each zone. Zones are described by the coordinated action of the 8$$ and 14**-19** arrays, the body descriptors. Each body is specified as a continuous volume bounded on left, right, inside, outside, bottom, and top by physical coordinates specified in the 14**-19** arrays. A zone number is associated with each body. If one envisions an array giving zone number by mesh cell, the effect of a body is to fill the enclosed portion of the array with the specified zone number. Each body overlays all previous bodies, so the final zone number for a cell is the last value specified for it. Body boundaries may extend beyond the system boundaries. If internal boundaries do not correspond to a mesh boundary, the closest mesh boundary will be selected. A close correspondence should be maintained in order to prevent surprises, however. It may be noted that additional intervals can be added within body boundaries without change to the input arrays, since they are in terms of physical dimensions.

Very complex zones can be specified in this manner, and the results are relatively easy to confirm. TORT produces an output map of the zone assignments in order to assist checking. This method of entry was suggested by the combinatorial geometry input used by Monte Carlo codes. In comparison, it is less powerful, but very simple to learn and apply. In general, TORT descriptions can be converted to MORSE[68] Monte Carlo descriptions if they are not too complex.

The 9$$ array assigns a cross section set to each zone, while the 7** array allows modification of material density in each zone.

### 5.2.6 Flux Sweep Formulations

TORT offers a choice of flux formulations: weighted-difference, nodal, or characteristic. All are designed to produce high accuracy in properly meshed problems free of numerical difficulty, and to produce plausible, non-negative results in the face of difficulty. The weighted-difference method is very old, tested and accepted. In comparison, nodal is new, but highly favored by theoreticians. Nodal costs more to solve a given mesh, but it can produce good results for very coarse meshes for which weighted-difference fails. The characteristic method is especially well-adapted to problems in which streaming through voids and dispersion around corners are important. It guarantees positive results for positive sources, and it is generally more accurate than the other positive methods. It operates faster than the nodal method on some workstations. Since it has not been vectorized, it is relatively expensive on a vector computer, however. Nodal and characteristic procedures are presently available only in XYZ geometry.

If weighted-difference is used, the traditional zero-weighted calculation can be performed by setting THETA=0. Otherwise, a theta-weighted calculation is done. WMAX can simulate older formulations. If WMAX = 1, the stable, but generally inaccurate, step calculation is simulated. If WMAX = 2, the negative-prone linear or diamond calculation is simulated.

If the original nodal formulation is used, WNODAL controls the treatment of the boundary shape distribution. WNODAL = 0 insures positive results but may damage accuracy at times. WNODAL = 1 allows a more aggressive, and presumably more accurate, treatment with some risk of negative contamination. WNODAL is not needed in the "positive nodal" formulation, and that formulation may prove to be preferable.

### 5.2.7 Directional Quadrature Set

Flux solution by the method of discrete ordinates requires a set of discrete directions and weights, the "directional quadrature set." Flux evaluation is performed only in these directions, and integrals over directional variables required in the transport formation are estimated by weighted sums. The direction sets consist of lists of weights and direction cosines input in the 81**-83** arrays. The directions are ordered according to increasing values of cosine with respect to the X axis (EMU), and then with respect to the Z axis (ETA). Thus, all downward directions precede all upward directions. EMU's with equal ETA are grouped together, and such a set is called an "ETA level." A third cosine, XZI, measured with respect to the Y axis, is required in the calculation, but it is calculated internally. After the user has supplied a sweep of EMU for each value of ETA in either the positive or the negative hemisphere, another identical sweep must follow it. The first will be assigned negative values of XZI, while the second will have positive values.

A weight, W, is given for each direction. Certain curved-geometry calculations require that each "ETA level" begin with a "boundary direction" for which XZI = 0 and W = 0. These boundary directions have no direct effect on XYZ calculations, but they must be supplied for compatibility.

### 5.2.8 Boundary Conditions

Boundary conditions must be supplied on each external surface to start the flux sweep in directions pointing away from that surface. The void B.C. is used at the edge of a system that is not surrounded by other material. It simply sets the incoming flux to 0. Reflection describes the action at a plane of symmetry, and it is often used at the center of a cylinder as well. With the reflective condition, incoming flux is set equal to the outgoing flux in the reflected direction. The cylindrical condition is best at the right boundary of a cell calculation, and it is often used at the center of a cylinder. With the cylindrical condition, each incoming flux is set to the average outgoing flux for that ETA level. If the problem is truly asymmetric in the theta direction, none of the conditions available are rigorously correct, however.

The albedo condition returns a fraction of the emerging current as an isotropic reflected flux. With an albedo of 1.0, the condition is a "white" boundary. The periodic condition at opposite boundaries describes a repeating pattern, e.g. an ABCABC pattern. It is the condition rigorously applicable to the theta boundaries when a full 360-degree problem is solved. A fixed value, i.e., a "boundary source," can also be specified.

All boundary conditions except void and fixed-source may slow flux convergence when applied to right, outer, or top boundaries. The partial-current rebalance acceleration does have special provisions to help with the convergence in such cases, however.

### 5.2.9 Fixed Source Input

External boundary sources are signaled by boundary conditions of 4. If NTBSI > 0, the sources are read from unit NTBSI and multiplied by CHS(*ig*) before use. If NTBSI = 0, a uniform source of CHS(*ig*) is supplied as a default. If NTBSI < 0, NTBSI is not used, but the source is constructed from arrays 91**-95**, which give it a separable variation with direction, space, and energy group. Each boundary uses only two of the three spatial shape arrays. For example, if IBB = 4 and IBT = 0, the k-boundary source array would be:

$$\text{FKO(m,i,j)} = \text{SHAPM(m)*SHAPI(i)*SHAPJ(j)*SHAPG(ig)} .$$

The upward-directed values would be used at the bottom boundary, as specified by IBB. The downward-directed values would be ignored in this example.

Internal k-boundary sources are signaled by a non-zero value of NKNTSR. The internal boundary source arrays are constructed from the 91**-93** arrays, giving them spatial shape, and they take their energy shape from CHS(*ig*):

$$\text{SORK}(m,i,j) = \text{SHAPM}(m)*\text{SHAPI}(i)*\text{SHAPJ}(j)*\text{CHS}(ig)$$

A different set of shape functions is read for each of the NKNTSR source planes.

If NTDSI > 0, then a complete distributed source description, including moment expansion, is to be read from NTDSI and multiplied by CHS(*ig*). If NTDSI < 0, then IABS(NTDSI) must point to a scratch unit on which the source generated from the 90+ arrays will be written. The spatial shape will be either the product of the 92**-94** arrays, or else the complete shape read in the 96** array, according to NEDSOR. In either case, the source is multiplied by a zone-dependent factor from the 91** array and an energy shape from the 95** array. A source generated in this method is always isotropic, and anisotropic sources require a user-supplied input file.

If XNF ≠ 0. and KTYPE = 0, all fixed sources are multiplied by XNF before use. The use of XNF with KTYPE ≥ 1 is described elsewhere.

It should be noted that the 90+ arrays that refer to the space mesh are likely to give unexpected results with the discontinuous mesh features, and the defaults should generally be used. Similarly, default values of the arrays referring to quadrature directions should be used with jobs having multiple quadrature sets.

### 5.2.10 Adjoint Ordering

If an adjoint problem is to be solved, all data in the first four input blocks are to be entered in the normal manner. All input files supplied by the user, as well as any arrays numbered 91** and higher, must be supplied in adjoint form, i.e., reversed with respect to energy. For files and arrays 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 $\mu > 0$, $\xi > 0$, 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 $(-\mu, -\xi, -\eta)$. Directional input files must be constructed accordingly, and directional output files must be so interpreted. Output files also have the energy groups ordered as calculated, i.e., ordered from lowest photon energy to highest neutron energy.

### 5.2.11 Flux Iteration Convergence

If the problem contains scattering from direction to direction within an energy group, the implicitness must be removed by iterations. Beginning with the best guess available, flux is evaluated repeatedly, the newest flux being used to update the scattering source between iterations. These flux iterations are continued until convergence is reached, i.e.:

$$\epsilon^P = \left| \frac{\Phi^n_{i,j,k,g} - \Phi^{n-1}_{i,j,k,g}}{\Phi^n_{i,j,k,g}} \right| W_{i,j,k} < EPP$$

where $n$ is the flux iteration index, and $W_{i,j,k}$ is the value of the importance function (20** array) for the position *ijk* if IEPSBZ > 0, and 1 otherwise. The largest $\epsilon^P$ for any group after the last flux iteration is called "group convergence", $\epsilon^G$.

### 5.2.12 Flux Iteration Acceleration

If the system is large and the scattering ratio is high, e.g., 0.8 or more scatters within the group for each particle interaction, flux iterations can be quite slow to converge. Synthetic acceleration is employed to speed the process. This method uses "acceleration iterations" or "rebalance iterations", which are, in effect, additional flux iterations conducted using a cheaper procedure. The acceleration iterations continue until convergence has been reached or until the acceleration limit, NSCLMX, has been reached. The optimum value of NSCLMX is both problem- and machine-dependent, but the default value, 50, suffices for many cases. The value of EPSCL allows the level of rebalance convergence to taper downward as the flux convergence nears its objective, saving CPU costs as compared to a fixed objective.

If the iteration limit is reached without convergence, and if convergence has reached CONACC, the improved data are accepted for use, and the number of acceleration iterations is reported as NSCLMX. If the convergence has not reached CONACC, the acceleration is deemed to have failed, the number of acceleration iterations is reported as negative, and unaccelerated data are carried forward to the next flux iteration.

### 5.2.13 Flux Acceleration Stabilization

The PCR method tends to overcorrect in certain situations, and its practical use requires a "damping factor." If CSOLIN > 0, the damping factor starts at a value of CSOLCN as each group is iterated, and it increases by CSOLIN with each successive iteration. CSOLCN should never be less than the default value, but larger values may be useful in very difficult problems. If CSOLIN < 0, the increment ABS(CSOLIN) will be applied whenever a set of internal tests determine the need. The use of CSOLIN < 0 allows less-skillful choices of CSOLIN to give good results. In spite of the judgemental nature of the damped PCR method, it is so successful in very difficult problems that many analysts will use nothing else.

### 5.2.14 Problem Type and Source Iterations

Several major types of calculation are available, chosen by KTYPE:

KTYPE=0: Fixed source calculation (in absence of fission), or subcritical multiplication (M search).

KTYPE=1: K search - determination of $k_{eff}$ for a subcritical or supercritical system.

KTYPE=2: A search - modification of a pure absorber concentration to produce a specified $k_{eff}$.

KTYPE=3: C search - modification of material concentrations to produce a specified $k_{eff}$.

KTYPE=4: D search - modification of dimensions to produce specified $k_{eff}$.

When KTYPE = 0 and NSRMX = 1, a simple fixed-source problem is solved. Flux from the input fixed source and fission source from an input flux guess, if any, are used to determine fluxes and a new fission density. If no fission or upscatter is present, the use of NSRMX > 1 is unnecessary and wasteful. If a system has fissile material, but the fission calculation is not desired, set the fission spectrum (CHI) to 0. In fact, CHI should be 0 in any calculation for which no fission is to be calculated, since that bypasses the allocation of two large arrays.

If upscatter is present, NSRMX > 0 must be used in order to allow source iterations to remove implicitness in the group-to-group scattering, whether fission is present or not. EPO > 0 should be used in this type of calculation, in order to ensure convergence of the flux between successive source iterations.

When fission is present and NSRMX > 1, a subcritical multiplication calculation (M search) will be conducted. After the first iteration, the secondary fission source is added to the fixed source to perform each successive iteration until all subsequent fission regenerations have been accounted for. When the process converges, the ratio of fixed+fission source to fixed source is the subcritical multiplication, M. If the system is very near critical, M may be 100 or more. If the system is supercritical, the search has no physical meaning, and it will fail.

When KTYPE = 1, a calculation of the fission multiplication factor, $k_{eff}$, is performed. The calculation starts with either a distributed source input or a flux guess from a previous problem. The fission resulting from the flux guess is added to the fixed source, if any, and the total source is normalized to XNF. Fluxes and the secondary source resulting from the fluxes are calculated. The $k_{eff}$ estimate from the iteration is the ratio of secondary source to the total primary source. Iterations are continued until $k_{eff}$ converges. The distributed source, if any, is ignored after the first iteration. A boundary source should not be used with this type of problem.

A relatively small number of flux iterations, often 2 or 4, are best with this type of calculation. If $k_{eff}$ is not close to 1, calculations can be improved somewhat be setting EVI equal to the best available estimate of K. This is especially important in the case of restarts from a well converged previous problem.

When KTYPE $\geq$ 2, properties of the system configuration are changed in order to drive $k_{eff}$ 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 sufficiently near the objective value. The user must assure that the configuration can be changed sufficiently to adjust $k_{eff}$ as required.

### 5.2.15 Source Iteration Convergence

In general, source iterations will be performed until the time limit, TMAX, is exceeded (IXX=8), the iterations limit, ISRMX, is exceeded (IXX = 1), or convergence is reached. The value of IXX is contained in the output monitor lines. Source iteration convergence has not been reached if:

IXX = 3: the indirect search has not converged, i.e. KTYPE ≥ 2, EPK > 0, and:

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

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:

$$\epsilon^F = \left| \frac{\Phi^N_{i,j,k,g} - \Phi^{N-1}_{i,j,k,g}}{\Phi^N_{i,j,k,g}} \right| > EPO \quad ,$$

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

$$\epsilon^D = \left| \frac{F^N_{i,j,k} - F^{N-1}_{i,j,k}}{F^N_{i,j,k}} \right| > EPF \quad ,$$

IXX = 6: $k_{eff}$ has not converged; i.e., EPK > 0 and:

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

IXX = 7: no source iterations have been completed.

The value of IXX is printed after each iteration, together with each of the above measures of convergence, except that, if EPO = 0, the corresponding convergence is not calculated.

If an additional iteration is required in order to obtain special output such as directional flux output, this is indicated by IXX = 9.

### 5.2.16 Indirect Searches

The indirect searches (KTYPE ≥ 2) 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 that adjusts $k_{eff}$ to an input value, EKOBJ. EV is used to adjust the system parameters as follows.

A search: $(A)_{eff} = (A)_{input}$*EV; A is the concentration of a pure absorber defined in an input array.

C search: $(C)_{eff} = (C)_{input}$*(1+$f$ (EV-1)); C is the concentration of one component in its mixture, defined in the 10$$ and 11$$ arrays, while $f$ is the component fraction in the 12** array.

D search: $(\Delta d)_{eff} = (\Delta d)_{input}$*(1+$f$ (EV-1)); d represents any space dimension and $f$ is the superzone search fraction input.

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:

$$CVK < EVTH*MAX(EPK,MIN(O.1,CONVSR)) .$$

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 ≠ 0, the second EV is:

$$EV = EVI +/- EVDELK ,$$

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 otherwise.

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

### 5.2.17 Source Iteration Acceleration

TORT uses an error-mode extrapolation to accelerate source iterations. The code accumulates the sum FERAH, an error-mode characteristic estimate $\lambda$:

$$FERAH_{new} = \sum \left| F^{N}_{i,j,k} - F^{N-1}_{i,j,k} \right| ,$$

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

and an error-mode removal parameter $\theta$:

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

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

$$F^{N+1/2}_{i,j,k} = F^{N}_{i,j,k} + \theta' \left[ F^{N}_{i,j,k} + F^{N-1}_{i,j,k} \right]$$

where $\theta'$ is $\theta$ adjusted so that no fractional change in fission density will exceed |SORMIN|, where |SORMIN| is an input value. The fluxes are similarly extrapolated by the factor $\theta'$, with each flux limited by the |SORMIN| ratio.

Tests show this procedure to be safe and effective in accelerating $k_{eff}$ calculations, especially those dominated by slow decay of a single error mode. It can also have an effect during upscatter calculations, discussed below. It is important to assure that enough source iterations are performed to allow calculation of a stable eigenvalue estimate, and that exactly the same number of flux iterations are performed in each source iteration.

### 5.2.18 Upscatter

If upscatter is present in a problem, upscatter rebalance is performed after the second source iteration and subsequent 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 IG $\geq$ IGUPS from IG $\leq$ IGUPS

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

The procedure would be less effective in very large systems where thermal spectra establish themselves in various regions relatively independently. It may be noted that, if the problem materials do not naturally have fission, adding a small amount of fictitious fission to a troublesome region may allow the source iteration acceleration to operate, speeding the convergence dramatically.

### 5.2.19 Subcritical Multiplication

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:

$$\text{FSNORM} = \text{FXSUM}/(\text{FXSUM}-(\text{SRNEW}-\text{FSOLD}))\ ,$$

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.. In problems where fission rescaling is effective, it can also be an important supplement to the error-mode extrapolation. For example, without fission rescaling, $\theta$, may be 100 or larger for a given problem. Single-precision calculations may not be sufficient to provide a stable value of $\theta$, and so the extrapolation is not effective. Rescaling may lower $\theta$ 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.

The user is cautioned that EPK is the criterion for the fractional change in one iteration, not an uncertainty estimate. A better estimate is $\theta$*EPK, which can be much larger.

### 5.2.20 Source Normalization

In fixed-source problems (KTYPE = 0), XNF is used as a multiplier of all fixed sources. In searches (KTYPE $\geq$ 1), normalization is such that the sum of the fixed source guess and the fission source resulting from the flux guess is equal to XNF. The fixed source guess is ignored after the first source iteration, of course.

For a given source, fluxes inherently tend to increase with increasing $k_{eff}$, and some speak of this as "fluxes normalized to $k_{eff}$." A more accurate statement is that the source input to an iteration is normalized to XNF, and so the fission calculated from the fluxes is larger by $k_{eff}$.

### 5.2.21 Key Flux Positions

In many cases, it may not be necessary or feasible to obtain flux information for all mesh cells in a problem. Accordingly, key flux positions can be specified by the 22**-24** arrays. Flux values at these key flux positions can be printed after each iteration or after all of the iterations for a group. If the positions do not correspond to the center of a mesh cell, the nearest value will be given. Response summaries, described below, can be obtained at the key flux positions.

### 5.2.22 Response Summaries

Arbitrary response functions can be folded with the group flux results and obtained as output at each key flux point, at each mesh cell, or as region integrals. If NTRSO $\neq$ 0, cellwise responses will also be written onto the file indicated. Each response is the product of a zone-dependent function and a group-dependent function entered as the 26**-27** arrays. If fluxes from a previous calculation are used without need for additional iterations, NSRMX can be set to 0, bypassing much of the setup operations, but still yielding the response data. In such a case, fixed-source information is not required or used.

### 5.2.23 Time Limit

If TMAX > 0, the problem will be discontinued when the time used exceeds TMAX. The interpretation of time usage is system-dependent, as discussed in the Environmental Information section. Time limit termination may occur before any source iteration, including the first, and before any flux iteration. The flux moment output file from a terminated problem will always contain the newest information applicable to a restart, but other output files may be incomplete. Problems terminated by system action can be very troublesome to restart, and the use of TMAX can avoid those difficulties.

### 5.2.24 User Error Code

When user errors, e.g., input errors, are discovered, an error level is assigned to each error and indicated in the corresponding message. A record of the highest error code encountered in the execution, NCND, is kept, and the execution tries to continue through the processing of all input data and files. If that processing is completed without some type of system failure due to the error, the condition is compared with an acceptance limit, NCNDGO, and the problem is stopped if the limit is exceeded. The user can alter the value of NCNDGO by the input of NCNDIN. A high value, e.g. 16, can allow the user to override unwanted error checks and proceed into calculation. Results of such a calculation should be regarded with suspicion, however.

### 5.2.25 Directional Flux Output

If the boundaries of a patch of the space mesh are entered in the 34** array, and if a unit is input as NTDIR, directional flux will be written onto the corresponding file for mesh cells within the patch. This will generally require one additional iteration, since it is not practical to save the directional flux data internally. It is generally helpful if the boundaries of the patch correspond to boundaries of the space mesh. This file may be very large, and caution in choosing the boundaries of the output patch is needed.

### 5.2.26 Discontinuous Mesh

If IM < 0, the 70+ arrays define a discontinuous space mesh. Programming details of this feature are discussed in Appendix D. In this case, the 2** and 3** arrays may contain more than one mesh boundary specification, one after the other. The 72$$ (74$$) gives the number of intervals in each I-set (J-set). The largest I-set should have length equal to the magnitude of IM, and the largest J-set should have length JM. Then, the 73$$ array assigns one J-set to each plane, while the 71$$ assigns one I-set to each row of each plane.

The purpose of this feature is to allow fine detailing of a special portion of the problem without requiring mesh boundaries to extend throughout the problem. When it is used, a coarse mesh must be defined for acceleration purposes, and each boundary of the coarse mesh must lie in every fine mesh set. This discipline reduces the cost of the acceleration, and it prevents certain distortions that can occur without it.

### 5.2.27 Variable Quadrature

If MM < 0, the directional quadrature is to vary with energy group. In this case, the 81**-83** arrays may contain more than one quadrature set specification, one after the other. The 76$$ array specifies the length of each quadrature set, while the 75$$ array assigns one set to each energy group. The largest set should have length equal to the magnitude of MM.

This feature allows a fine direction set to be used in high-energy groups, in which it is often necessary to calculate the transport of particles away from strong local sources. Lower groups, where scattering often dominates, may be adequately solved with a coarse quadrature. This feature can save execution time and can reduce the size of certain files.

### 5.2.28 Variable Cross Section Expansion

If MM < 0, the 77$$ array may be used to specify the order of Legendre expansion of the flux moments by energy group. The high-energy groups may benefit from a high-order expansion in certain problems for reasons analogous to the need for fine quadrature. The use of a lower-order expansion in lower energy groups can save both time and file space. The usual restrictions on

expansion order for a particular quadrature set apply here.

### 5.2.29 Accurate Restart Procedure

When some groups in a previous run have converged satisfactorily, additional flux iterations in those groups are not needed unless additional source iterations are to be done. These can be bypassed by adjusting the 21$$ array, but that is not always convenient. Instead, if NIFCNV is set to 1, information on the flux restart file will cause unnecessary iterations to be bypassed.

The Damped Partial Current Acceleration procedure typically increases the amount of damping as the problem progresses, in order to achieve best convergence. When a problem is restarted, however, the damping progression starts again unless it is otherwise regulated, and the total number of iterations required for convergence may be increased. If 10 is added to the value of NIFCNV, the damping will be initialized from information on the flux restart file, rather from the input control parameters.

### 5.2.30 Direct Access Scratch File Retention

If NTFOG < 0, the direct-access scratch files containing flux moments (unit 91), as well as boundary source and flux directional data (unit 94) will be retained at the end of the execution. In this case, only a small amount of control information will be written on the file pointed to by NTFOG, and the necessity of duplicating the flux moments in sequential form on NTFOG will be avoided. This can reduce the total amount of file space required, and it reduces the I/O work. It clearly requires that the storage mode be such that flux moments are stored on the scratch file, and this should be enforced by using MINBLK ≥ 1.

To transfer the direct-access files, together with the small sequential file, to a new problem, simply tag the number of the sequential file with a negative in specifying NTFLX. It is important that the second problem be identical to the first in details such as the sources, mesh, number of groups, boundary conditions, and the like. The number of iterations per group and the time limit may be changed as desired, however.

It may also be desirable to input boundary directional source information on unit 94, avoiding the requirement for duplicate copies of this potentially large file. For this option, set NTBSI=0, and set NIFBND = 1. In this case, the distributed source information will be read directly from unit 94, which must have been written previously by a preprocessor code. This operation carries the restriction that the file segment size, LCMOBJ, must be identical to that used in the preprocessor code.

## 5.3 PROBLEM PRINTED OUTPUT

Each TORT 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, the array number and length are noted.

The input parameter arrays (61$$-67**) are edited, together with a brief explanation of each. Region, material, density, importance, iteration limit, and energy spectral arrays follow (87$$, 9$$, 7**, 20**, 21$$, 1**, and 6**). The directional quadrature edit is next, with W, EMU, and ETA from input arrays (81**-83**), followed by related data determined internally: the third direction cosine (XZI), the directional coupling array (BETA), and the direction mates. The EMU, XZI, and ETA mates indicate the number of the direction corresponding to reflection in the given direction cosine. A particle moving in direction m, for example, upon reflection at a Z boundary, would move away in the direction given by ETA MATE(m). The LEVEL MATE has the value of the number of the first direction of the "eta level" in which direction m lies; i.e., the first of the set of contiguous directions with common ETA's and with XZI's of the same sign. The first mate of each level is tagged with a negative.

Space-mesh information follows: the K- and J-boundary arrays (4** and 3**), together with mesh midpoints, and interval widths. Each of these arrays is accompanied by the number of the coarse-mesh interval corresponding to each fine-mesh interval. In addition, the K-boundary edit includes the number of the J-mesh corresponding to each plane and the boundaries of the distributed flux output patch, if any. The next edit gives the I-boundaries (2**), together with midpoints, interval widths, coarse-mesh interval numbers, and three arrays, AH, AV, and VR used in the flux solution. The J and I edits are repeated, once for each mesh set, if the discontinuous mesh feature is used.

The key flux locations (22**-24** arrays), the mesh intervals corresponding to the key fluxes, and the coarse mesh boundaries (84**-86**) follow. The next edit gives the body array information (8$$ and 14**-19** arrays), followed by a summary of fine- and coarse-mesh information.

A zone map gives the zone layout determined from the body description. An edit of storage usage (generally of value only to the programmer) indicates the end of key storage areas. This edit may be given as many as four times as the code attempts to adapt to the available memory by using more external storage. An estimate of disk space required for the scratch files (81-95) follows. An optional edit of cross sections is valuable in diagnosing trouble. If NEDSOR has called for an edit of the distributed source, this follows next. An edit of unnormalized total fission source and groupwise distributed source is given, and then the monitoring of the iteration process begins.

A source iteration monitor line gives initial values before the first iteration, and those values are updated after each subsequent source iteration. Entries on this monitor line are:

ITN Source iteration number
FLX Number of flux iterations completed for this source iteration
CUM Cumulative number of flux iterations

CNV 10*IXX + IFCONV

| | | |
|---|---|---|
| IFCONV | = 0 | Source iterations are to continue |
| | = 1 | One more iteration is required for special output |
| | = 2 | Iterations are to stop at once |
| | = 3,4,5 | As in 0,1,2 but configuration recalculation is required |

(IXX was explained earlier; configuration recalculation is used in indirect searches.)

ACC 100*ISR + 10*NEWCFG + IACX

| | | |
|---|---|---|
| IACX | = 0 | Extrapolation of fluxes was bypassed |
| | = 1 | Extrapolation of fluxes was performed |
| NEWCFG | = 0 | Old configuration is to be used |
| | = 3 | New configuration is to be used |
| ISR | = 0 | No eigenvalue extrapolation was performed |
| | = 1 | Linear eigenvalue extrapolation was performed |
| | = 2 | 2-point nonlinear eigenvalue extrapolation was performed |
| | = 3 | 3-point nonlinear eigenvalue extrapolation was performed |

(IACX applies to $k_{eff}$ calculations. IACX, NEWCFG, and ISR apply to indirect searches.)

| | |
|---|---|
| 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 |
| FSN-CNV | Fission convergence |
| FLX-CNV | Flux convergence |
| GRP-CNV | Group convergence |
| SRH-CNV | Search convergence |
| UPS-CNV | Upscatter convergence |

| | |
|---|---|
| EXTRAP | Fission extrapolation factor in error-mode extrapolation process |
| TIME | Time iteration was completed |

EV and K are interpreted as follows:

$$EV = \sum_g V \nu_g \sigma_g^f \phi_g \quad , \text{(non-adjoint)}$$

$$EV = \sum_{g'} x_{g'} V \phi_g \quad , \text{(adjoint)}$$

$$K = \sum_{g'} x_{g'} \sum_g V \nu_g \sigma_g^F \phi_g \quad .$$

If the $\chi$'s do not sum to 1, K will differ from EV in a K-calculation by a ratio equal to the sum of the $\chi$'s . Even if $\chi$ is precisely normalized to 1.0, 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 |
| KMFD | K-position of maximum flux change |
| MX FX DV | Maximum relative flux change in this iteration |
| MX DV FX | Most rapidly changing flux value |
| REBL | Number of rebalance iterations; a negative value means the rebalance results were not accepted for use; a number larger than NSCLMX means that the acceleration was not successfully converged to CONSCL, but the results satisfied the CONACC criterion, and were accepted for use. |
| REBL ERR | Largest relative change in accelerated values in the last rebalance iteration |
| MAX REBL | Maximum relative change due to acceleration |
| RB DV FX | Relative change in MX DV FX due to acceleration |
| GRP REBL | Average relative rebalance change for the entire group |
| KEY FLUX | Value of first key flux requested by input data, after acceleration |

NEG FIX — Maximum fraction of total source found to be negative in any mesh cell. This indication will be set to 1.0 if the "economy fixup" is used, since the calculation of a true value would be too costly.

SOURCE — Total of all sources into a group, including external boundary sources. This will be zero for iterations in which no acceleration is done.

TIME — The time at which the group iteration is started is given at the end of the heading column.

Arbitrary diagnostic information is given below the time indication, and it is to be ignored by the user.

If IEPSBZ > 0, an edit of pointwise flux convergence by region is given, either after each flux iteration or after the last. Similarly, key flux values are printed according to NKEYFX. If IFXPRT = 1, scalar fluxes will be edited after the last iteration. If IFXPRT = 2, the scalar flux values are edited for each group after each iteration. This feature may be useful in preliminary testing of a new problem setup, for example, but it is obviously impractical for large calculations having many source iterations. It should be used with care.

The key response edit, controlled by NRESP, follows. Pointwise values at the key flux positions and region integral values are always given. Pointwise values in each mesh cell may be printed, optionally.

If ITALLY = -1, the problem ends with a summary of time usage by each major code section. The code sections tabulated include:

| | |
|---|---|
| MSC | All sections not listed elsewhere; mainly editing and file processing. |
| I/O | Actual transfer of data to and from external files. |
| SOURCE | Preparation of the "in-source" into a group. |
| FLUX | Calculation of the "total-source" for a group. |
| PLANE | Application of boundary conditions; coordination of flux solution; testing of flux convergence. |
| ROW | Actual discrete ordinates flux sweep. |
| CMSCLR | Acceleration. |
| SORX | |
| SNPL | |
| WWESOL | |

Non-labeled columns, for temporary use of the programmer, follow. The exact interpretation of the various times is discussed in the Environmental Information section.

## 5.4 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).[66,67] The specifications of these formats are contained in Appendix B. The files are:

| FILE NAME | FORMAT | USE |
|---|---|---|
| NTFLX | FLXMOM | Flux and moment input |
| NTFOG | FLXMOM | Flux and moment output |
| NTSIG | GIP | Cross section input |
| NTBSI | VARBND | Boundary source input |
| NTDSI | FLXMOM | Distributed source moment input |
| NTRSO | FLXMOM | Cellwise response output |
| NTZON | VARMAP | Zone map input |
| NTDIR | DIRRAW | Directional flux output |

## 5.5 SCRATCH DATA FILES

Appendix B also includes definitions for two of the internal scratch formats, since these can be transferred between problems. Scratch files using these formats are:

| UNIT | FILE | FORMAT | USE |
|---|---|---|---|
| 91 | NDFLX | TORSCRF | Flux Moments, All Groups |
| 92 | NDSOR | " | Total Source for One Group |
| 93 | NDSIN | " | In-scatter for One Group |
| 94 | NDBFX | TORSCRS | Boundary Sources + Fluxes |

The length of each file is specified by the formats, with the understanding that NDSOR and NDSIN contain data for only one group. It should be noted that the I/O routines may increase the disk space requirement by rounding small records to a full block.

Some files, intended exclusively for internal use, do not use defined formats. These include:

| UNIT | FILE | USE |
|---|---|---|
| 81 | NDKSR | K-boundary distributed source |
| 82 | NDSIG | Cross-section scratch |
| 83 | NDBTL | Scalar flux for edits |
| 84 | NDZON | Zone map |

| | | |
|---|---|---|
| 85 | NDCPC | Damped partial currents rebalance SOR |
| 95 | NDFIJ | Old scalar flux |

The record lengths and counts for these files are as follows:

| FILE | RECORD LENGTH | RECORD COUNT |
|---|---|---|
| NDKSR | MM*IM*JM | NKNTSR |
| NDSIG | IHP*MTM | IGM |
| NDBTL | IM*JM*KM | IGM |
| NDZON | IM*JM | KM |
| NDCPC | ICM*JCM | 8*KM |
| NDFIJ | IM*JM*KM | IGM |
| | IM*JM | KM |

**NOTES:**

IHP = IHM + 1 if IUPS > 0; else IHP = IHM

NDFIJ has two types of records, as indicated. Records of the first type are present only if EPXTR ≠ 0. and NSRMX > 1, or if EPO ≠ 0. Records of the second type are always present.

# Table 5.1. Detailed Array Input Requirements

TORT Parameter Arrays

```
61$$  input/output file logical unit numbers - - -

 ntflx  =  flux guess input unit          if .gt. 0
 ntfog  =  flux output unit               if .gt. 0
 ntsig  =  cross section unit                       (default=8)
 ntbsi  =  boundary source input unit     if .gt. 0 (neg: input from 90+ arrays)
 ntdsi  =  distributed source input unit  if .gt. 0 (neg: scratch unit; input from 90+ arrays)

 ntrso  =  response output unit           if .gt. 0 (neg: formatted output)
 ntnpr  =  large-scale output unit        if .gt. 0
 ntscl  =  scalar flux output unit        if .gt. 0
 ntzon  =  zone map input unit            if .gt. 0
 ntdir  =  directional flux output unit   if .ne. 0 (neg: write reduced-size file)

 ntcnvg =  convergence data output unit   if .ne. 0 (neg: convergence edit also provided)


62$$  control integers - - -

 nsrmx  =  source iteration maximum                  (default=1)
 nfxmx  =  flux iteration max. per group per itn.    (default=20; neg: limit by group in 21$$)
 icsprt =  0= cross sections not printed; 1= printed
 ifxprt =  0= scalar flux not printed; 1= printed at end; 2= printed as calculated
 modep  =  0= no effect; 1= bypass flux iteration print

 nkeyfx =  length of key flux array                 (neg: print key fluxes last iteration only)
 iepsbz =  0= no effect; 1=use zone importance; 11= 1+print cnvg. final itn.; 21= 11+print cnvg.
             every itn.
 nresp  =  no. response functions                    (neg: cell responses not printed)
 ktype  =  0= fixed source; 1= k-search
 nsctt  =  maximum order of flux expansion           (default=-1; neg: use nsctm)

- 10 - - -

 iadj   =  0= forward solution
 negfix =  0= no effect; 1= full source fixup; 2= initial fixup; 3= tested fixup; -1= economy
             fixup
 nsclmx =  acceleration iteration limit              (default=50)
 nbfacc =  flx itns. before acceleration             (0 ignored)
 mode   =  flux sweep: 0=opt wtd; 1= alt wtd; 2= opt xyz nod; 3=alt nod; 4=opt new nod; 5=alt
                       new nod; 7=characteristic

 nbuf   =  use 0
 locobj =  initial memory allocation, words*1000 (0 implies use default)
 lcmobj =  file segment size, words*1000 (0 implies unlimited segment length)
 minblk =  minimum no. of k blocks per group        (0: all groups stored in memory if possible)
 maxblk =  maximum no. of k blocks per group         (0: maximum is km)

- 20 - - -

 ncndin =  maximum user condition code               (default=4)
 ingrps =  no. groups of flux guess avail.           (0: all groups available; neg: no groups
           available)
 inmoms =  no. moments of flux guess avail.          (default=-1; no effect)
 indirs =  0=no effect; 1=ignore boundary flux guess
 nintsr#=  no. i-boundary internal sources used
```

```
 njntsr#=  no. j-boundary internal sources used
 nkntsr =  no. k-boundary internal sources used
 nintfx =  no. i-boundary internal fluxes saved
 njntfx =  no. j-boundary internal fluxes saved
 nkntfx =  no. k-boundary internal fluxes saved

- 30 - - -

 iacc   =  use 0
 itally =  0: no effect; -1: internal timing analysis
 ncpu   =  0: no effect; > 0: # DP tasks in Cray macrotasking environment
 nedsor =  0: 3-d source description; 1: 1-d; +10: zone*group edit; +100: cellwise edit
 nifcnv =  0: no effect; 1: skip cvgd grps on ntflx; +10: use rebal damping from ntflx

 nifbnd =  0: default bndy source if ntbsi=0; 1: find bndy source on direct access file if
           ntbsi=0


63$$  geometry controls - - -

 im     =  maximum number of x (or r) intervals (neg: number varies with row and plane)
 jm     =  maximum number of y (or theta) intervals
 km     =  maximum number of z intervals
 ibl    =  left    b.c.              (0=void         )
 ibr    =  right   b.c.              (1=reflected    )

 ibi    =  inside  b.c.              (2=periodic     )
 ibo    =  outside b.c.
 ibb    =  bottom  b.c.              (4=bndy source  )
 ibt    =  top     b.c.
 ingeom =  0= x-y-z geometry; 1= r-theta-z; 10= x-z; 11= r-z; 20=x-y; 21=r-theta

- 10 - - -

 izm    =  number of material zones
 inreg  =  use 0
 mm     =  maximum number of directions in quadrature (number varies with energy group)
 mmesh  =  number of material zone bodies
 spare  =  use 0

 nism   =  standard i-set for boundary flux (if.gt.0)
 njsm   =  standard j-set for boundary flux (if.gt.0)
 spare  =  use 0


64$$  cross section controls - - -

 igm    =  number of energy groups
 ncrx   =  number of materials
 nsctm  =  maximum order of scattering expansion
 iht    =  number of principal c.s. per group
 ihm    =  total number of scalar c.s. per group

 iups   =  number of upscatter c.s. per group
 neut   =  last neutron group                (0 if all groups are photons)
```

```
66** control reals - - -

 epp    =  pointwise flux convg. crit.          (default=1.0e-3)
 epf    =  fission convg. crit.                 (default=1.0e-3)
 epo    =  source iteration flux convg. crit.   (default=0.)
 epk    =  eigenvalue convg. crit.              (default=1.0e-4)
 spare  =  use 0

 conscl =  min. acceleration itn. convg. crit. (default=1.0e-4)
 conacc =  max. acceleration acceptance crit.  (default=1.0e-1)
 csolcn =  acceleration damping constant       (neg: use only on first source iteration;
           default=1.5)
 csolin =  acceleration damping increment      (neg: use only as required; default=-1.5)
 csolmn =  use 0

- 10 - - -

 flxmin =  minimum flux for convg. tests       (default=1.0e-30, 1.0e-60 on cray)
 theta  =  source multiplier, weighted difference mode (default=0.9)
 wmax   =  0=no effect; 1=weighted difference constrained to step; 2=constrained to linear
           (diamond)
 wnodal =  0=no negative risk, nodal mode; 1=higher risk
 sorneg =  minimum source neg. content repaired (default=1.e-3)

 evi    =  initial eigenvalue                  (default=1.)
 evdelk =  initial eigenvalue increment        (default=0.3)
 devdki =  initial eigenvalue slope guess      (default=-1.)
 evchm  =  maximum ev change ratio, each itn.  (default=1.5)
 evmax  =  maximum ev change ratio, overall    (default=10.)

- 20 - - -

 evkmx  =  maximum valid (keff-ekobj)          (default=1.)
 evth   =  keff convergence ratio              (default=0.2)
 epxtr  =  keff extrapolation convg. criterion (default=0.2; neg. bypasses subcritical
           rebalance)
 fisorf =  use 0
 epscl  =  ratio accel. convg. to flux convg.  (0 ignored; default=0.1)

67** problem parameters - - -

 xnf    =  source multiplier                   (0 ignored)
 tmax   =  maximum cpu min. for this problem   (0 ignored)
 ekobj  =  keff sought in search               (default=1.)
 eki    =  initial keff estimate               (0 ignored)

# = feature not completed
```

```
Parameter Input Arrays
  61$  input/output file logical unit numbers
  62$  procedure control integers
  63$  geometry control integers
  64$  cross section control integers
  66*  procedure control reals
  67*  problem control reals

Primary Dimension Setting Arrays
  71$ iset    i-set by row and plane                    (jm*km values; default=1)
  72$ imbis   # of cells by i-set                       (jm*km values; default=iabs(im))
  73$ jset    j-set by plane                            (km values; default=1)
  74$ jmbjs   # of rows by j-set                        (km values; default=jm)
  75$ mset    m-set by energy group                     (igm values; default=1)
  76$ mmbms   # of directions by m-set                  (igm values; default=iabs(mm))
  77$ nsctg   order of flux expansion by energy group   (igm values; default=nsctt)
    notes:  71$ - 74$ arrays are required only if im<0
            71$, 72$, and 74$ are usually longer than required;  fill arrays with 0
            75$ - 77$ arrays are required only if mm<0

Secondary Dimension Setting Arrays
  81* w       quadrature weights                        (mms values for each quadrature set)
  82* emu     first-direction cosines                   (mms values for each quadrature set)
  83* eta     last-direction cosines                    (mms values for each quadrature set)
  84* rcmb    x/r coarse mesh boundaries                (ima+1 values, then fill with large
              number)
  85* thcmb   y/theta coarse mesh boundaries            ( jm+1 values, then fill with large
              number)
  86* zcmb    z coarse mesh boundaries                  ( km+1 values, then fill with large
              number)
  87$ iznrg   region number by zone                     (izm values; default=1,2,3...)
    notes:  mms = number of directions in quadrature set; ima = iabs(im)
            if 84*, 85*, or 86* is omitted, it defaults to the fine mesh values
            if im<0, each coarse mesh boundary must lie in each fine mesh
```

General Input Arrays

| | | |
|---|---|---|
| 1* chi | fission energy spectrum | (igm values) |
| 2* rin | x/r fine mesh boundaries | (ims values for each i-set) |
| 3* thin | y/theta fine mesh boundaries | (jms values for each j-set) |
| 4* zin | z fine mesh boundaries | (km values) |
| 5* ener | energy group boundaries | (igm+2 values:upper bounds for groups 1,..,igm + lower bound for group igm + lower bound for group neut; 0 if neut=0) |
| 6* chs | fixed source energy spectrum | (igm values; default 1.) |
| 7* dnij | material density by zone | (izm values) |
| 8* inzn | zone # by body | (mmesh values) |
| 9* izmt | material # by zone | (izm values) |
| 14* gemx1 | body left boundaries | (mmesh values) |
| 15* gemx2 | body right boundaries | (mmesh values) |
| 16* gemy1 | body inside boundaries | (mmesh values) |
| 17* gemy2 | body outside boundaries | (mmesh values) |
| 18* gemz1 | body bottom boundaries | (mmesh values) |
| 19* gemz2 | body top boundaries | (mmesh values) |
| 20* epsbz | convergence importance by zone | (izm values) [if iepsbz>0] |
| 21* itmbg | iteration limit by group | (igm values) [if nfxmx<0] |
| 22* ceyai | i- key flux locations | (nkeyfx values) |
| 23* ceyaj | j- key flux locations | (nkeyfx values) |
| 24* ceyak | k- key flux locations | (nkeyfx values) |
| 26* dnres | zone response by zone, then by response | (izm*iabs(nresp) values; default=1.) |
| 27* csres | group response by group, then by response | (igm*iabs(nresp) values; default=1.) |
| 30* kntsr | locations of k-boundary internal sources | (nkntsr values) |
| 34* pchbn | boundaries of the dir. flux output patch | (8 values) |

notes: ims = number of cells in a row
jms = number of rows in a plane

```
External Boundary Source Arrays  [if ibl,ibr,ibi,ibo,ibb or ibt=4]
  91* shapm   source shape by direction                (mm  values; default=1.)
  92* shapi   source shape by i                        (im  values; default=1.)
  93* shapj   source shape by j                        (jm  values; default=1.)
  94* shapk   source shape by k                        (km  values; default=1.)
  95* shapg   source shape by group                    (igm values; default=1.)
    note:  these arrays may give unexpected results if im<0 or mm<0

Internal Boundary Source Arrays  [nkntsr blocks if nkntsr>0]
  91* shapm   source shape by direction                (mm  values; default=1.)
  92* shapi   source shape by i                        (im  values; default=1.)
  93* shapj   source shape by j                        (jm  values; default=1.)
    note:  these arrays may give unexpected results if im<0 or mm<0

Distributed Source Arrays  [ntdsi<0]
  91* shapm   source shape by region                   (nreg values; default=1.)
  92* shapi   source shape by i                        (im   values; default=1.)
  93* shapj   source shape by j                        (jm   values; default=1.)
  94* shapk   source shape by k                        (km   values; default=1.)
  95* shapg   source shape by group                    (igm  values; default=1.)
  96* flij    distributed source distribution by i,j,k (im*jm*km values)
    note:  these arrays may give unexpected results if im<0 or mm<0
           nreg = max{iznrg(i),i=1,izm}
           if mod(nedsor,10)=0, the source spatial shape is defined by the 96* array.
           otherwise, it is defined by the product of 92*, 93*, and 94* arrays.
           in either case, the spatial shape is multiplied by the 91* and 95* arrays.
```

# REGION 6. ENVIRONMENTAL INFORMATION

## 6.1 UNLOADING AND INSTALLATION OF THE DISTRIBUTION FILES

TORT and DORT are now distributed via a tape archive file. This archive also contains several utility programs used to maintain the software. CMP is an UPDATE emulator developed at Sandia National Laboratories. RSCORS is a graphics library also developed at Sandia. Most if not all of these utilities are installed during the DORT and TORT installation.

To install DORT and TORT, create a new directory (or choose an existing directory), then change to that directory. Now copy the files from the installation tape, e.g.,

```
mkdir   ~/doors
cd      ~/doors
tar xvf  /dev/rmt0      (use your correct tape device)
```

Change to the Install subdirectory and make the bin and current directories part of the search path, i.e.,

```
cd  Install
export  PATH=.:$HOME/doors/bin:$PATH   (if using ksh); or
set  path = (. $HOME/doors/bin $PATH)     (if using csh)
```

Then enter sinstall.unx and respond to the prompts. Detailed installation instructions are provided in Appendix C.

## 6.2 THE JDOS DRIVER

The JDOS procedure uses the DRV code, created by the same procedure that installed GIP, etc, to solve problems that require a sequence of various codes. This driver evolved from the Discrete Ordinates System (DOS)[6] that has been in use since the mid 1970's. (Unfortunately, the acronym for this system was later used for an entirely unrelated product by another provider of software.) A typical input stream might appear as follows:

```
=gip
[input data for GIP]
=tort
[Input data for TORT]
=end
```

The procedure provides an edit of the entire input stream, and it then invokes each code as required until the "=end" is reached. Execution is stopped after an abnormal failure of a code application.

## 6.3 INTERNAL MEMORY

On each system on which these codes have been demonstrated, it has been possible to install a run-time memory allocation. This means that allocation of the large "container array" in which most working data are kept can be delayed until the actual length requirement can be determined. This requirement is different for each problem, and it is generally not desirable to allocate a container sufficient for the largest possible problem routinely. The best procedure for each machine is selected by the mudort and mutort scripts. To replace this machine-dependent package with an ordinary FORTRAN fixed-dimension routine, the dort/src/mudort.{sys} file must be modified.

The amount of memory required and its arrangement are discussed in the Programmer's Information region. A special feature that might be noted here is the "direct access user buffer" area. In the largest problem solutions, the flux and source moment files are the target of much of the I/O activity. These files are direct access files, written and read in blocks as large as the memory allows. Data are moved to and from the files through the user buffer area, an area at the end of the user memory. Information can be requested from the buffers in a method convenient to the using program, and a special traffic manager routine locates the desired strip of data. The data in the buffer can be accessed in any order, read or written repeatedly as required, and then moved back to the external file when another buffer is required.

In addition to simplifying the programming, this arrangement allows the user to take advantage of any special large memory area such as those used on CDC and IBM machines of a few years ago. We do not provide procedures for doing this on present-day machines, however.

The DORT and TORT codes have various modes of tradeoff between memory size and amount of I/O activity, as discussed in the Programmer's Information region. If virtual memory were transparent in cost and speed, as it is in programming style, the best efficiency would be obtained by making the internal container array as large as required to minimize the I/O activity. In practice, the delays incurred with virtual memory on workstations have made it advisable to ensure that the memory allocation is of such a size as to remain in memory at all times. Unfortunately, that means that, if a second large problem is dropped in beside the first, both are likely to experience unsuitable delays as the paging rate becomes large. It is sometimes possible to monitor the paging rate, but this requires a high level of system familiarity and user attentiveness.

## 6.4 EXTERNAL FILES

Standard UNIX naming conventions are followed for files; logical unit number xx is assigned the name fort.xx, and xx must be 99 or less. Three unit numbers are reserved for special use:

- 5 Standard input,
- 6 Standard output, and
- 7 Simulated "punch" output.

User files supplied to the codes or obtained as output are generally of the sequential type, and they may have numbers [1-4] and [8-80]. The contents and formats of these files are discussed in the User's Information section.

Scratch files are assigned numbers 81-99. The contents and formats of these are also discussed in the User's Information section. Files 81-90 can be processed by either sequential- or random-access commands. Although the codes are initially set up to use random access, comments in the VARIO subroutine indicate how to change this to sequential access if it is preferred on a given system.

Files 91-93 store flux and source moment data, communicated through the user buffers described earlier. Files 94-99 transfer data directly to and from their arrays in memory, interrupted only by system buffers, if any. These files must use direct access.

In normal operation, restarting an incomplete problem is accomplished by writing a sequential copy of the flux moment file and, optionally, the boundary flux file. This file can also be used as a guess for a subsequent problem. It does require that two copies of the information exist at one time, however, and that can challenge the capacity of even the largest computing systems when the largest problems are run. Because of this, a special input option allows the user to save files 91 and 94, which contain the required information. Since these are scratch files, they do not contain traceability information, and the user must perform the necessary record keeping.

For very large problems, finding disk space for the largest scratch files can be difficult, and storing them on an off-line medium can be impractical or impossible. The LCMOBJ parameter causes the files to be created and used in segments of a convenient length.

## 6.5 SPECIAL FEATURES

As discussed in the Programmer's Information region, communication with special system library subroutines is done through a machine-dependent interface package. This package handles low-level I/O operations, communication of time, date, and user information, special services at the beginning and ending of execution, and the like. A generic substitute package is discussed in the comment files, and it can be used if required.

Where possible, each problem provides various types of time summaries. A monitor line appears after each major section, giving "accumulated charge" and "charge increment". This charge is the sum of CPU time and, where available, system time used. It is also printed on the monitor line before and after the solution of each energy group. At the end of the problem, the charge is broken down by CPU and system component. The Cray and IBM installations presently provide both CPU and system time information, while others do not. On mainframe systems, the operating system is likely to enforce a time limit, and the input allows a maximum charge to be entered. After using this amount of time, the execution will be terminated as quickly as possible.

Real time is also printed at the end of each problem. If a high-precision real-time clock is available, it is used. Otherwise, the internal time-of-day indication is decoded to provide data accurate to the nearest second.

The system library routines are typically written in C language, and another C-language routine is often required to couple the information into the FORTRAN code. The appropriate routines are selected by the installation procedure.

On Cray systems, an optional Cray Assembler Language (CAL) package is provided for use with weighted difference flux calculations. If this is troublesome, it can be removed, and default FORTRAN routines will take its place. A penalty in execution speed as large as a factor of three may be observed, however. The input must ask for the alternate (scalar) flux mode in that case. A warning message has been inserted to prevent the use of the vector mode without the CAL package, since this is unacceptably inefficient.

## 6.6 COMMENT FILES AND DEMONSTRATION PROBLEMS

Several comment files in the dort and tort subdirectories are created as the codes are unloaded. It is important that these be read. One of the files instructs the user as to how to run the DORT and TORT demonstration problems. Typical running times experienced in routine use, without special fine tuning, are given. In another file, a discussion of the problems is given. A new installation should always include solving the problems and comparing the results with those distributed with the source material.

## 7. REFERENCES


1. W. A. Rhoades and R. L. Childs, "An Updated Version of the DOT 4 One- and Two-Dimensional Neutron/Photon Transport Code," ORNL-5851 (July 1982).

2. K. D. Lathrop, "THREETRAN: A Program to Solve the Multigroup Discrete Ordinates Transport Equation in (x,y,z) Geometry," LA-6333-MS (May 1976).

3. T. Nishimura, K. Tada, and H. Yokobori, "Development of Discrete Ordinates $S_n$ Code in Three-Dimensional (X,Y,Z) Geometry for Shielding Design," *Nucl Sci. Tech.* **17**, 7, 539 (July 1980).

4. W. W. Engle, Jr., "ANISN, A One-Dimensional Discrete Ordinates Transport Code with Anisotropic Scattering," K-1693 (March 1967).

5. D. R. Vondy, T. B. Fowler, and G. W. Cunningham, "VENTURE: A Code Block for Solving Multigroup Neutronics Problems Applying the Finite-Difference Diffusion-Theory Approximation to Neutron Transport, Version II," ORNL-5062/R1 (November 1977).

6. W. A. Rhoades and M. B. Emmett, "DOS: The Discrete Ordinates System," ORNL/TM-8362 (May 1982).

7. G. I. Bell and S. Glasstone, *Nuclear Reactor Theory,* Van Nostrand Reinhold, 1970.

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

9. L. Boltzmann, *Vorlesungen uber Gastheorie,* Leipzig: Johann Ambrosius Barth, 1910.

10. B. Davison, *Neutron Transport Theory,* Oxford University Press, 174, 1957.

11. S. Chandrasekhar, *Radiative Transfer,* Oxford University Press, 1950.

12. G. C. Wick, "Uber Ebene Diffusionsprobleme," *Z. Phys.* **121**, 702 (1943).

13. 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).

14. B. G. Carlson and K. D. Lathrop, "Transport Theory - The Method of Discrete Ordinates," LA-3251-MS (Rev. October 22, 1965).

15. F. R. Mynatt *et al.*, "Development of a Two-Dimensional Discrete Ordinates Transport Theory for Radiation Shielding," CTC-INF-952 (August 1969).

16. 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).

17. 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).

18. T-1 Division, "Modified Version of DTK," Informal Los Alamos National Laboratory Memorandum (November 1962).

19. 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).

20. W. W. Engle, Jr., F. R. Mynatt, and R. S. Booth, "One-Dimensional Time-Dependent Discrete Ordinates," *Trans. Am. Nucl. Soc.* **12**, 400-401 (June 1969).

21. 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).

22. K. D. Lathrop, "User's Guide for the TWOTRAN (X,Y) Program," LA-4058 (February 1969).

23. K. D. Lathrop, "Spatial Differencing of the Transport Equation: Positivity vs. Accuracy," *J. Comput. Phys.* **4**, 475 (1969).

24. W. A. Rhoades and F. R. Mynatt, "The DOT III Two-Dimensional Discrete Ordinates Transport Code," ORNL/TM-4280 (September 1973).

25. 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).

26. R. L. Childs and W. A. Rhoades, "Theoretical Basis of the Linear Nodal and Linear Characteristic Methods in the TORT Computer Code," ORNL/TM-12246 (January 1993).

27. E. R. Cohen, "Some Topics in Reactor Kinetics," *Proc. Second Int. Conf. on Peaceful Uses of Atomic Energy (PUA)* **11**, 302-309 (1958).

28. E. R. Cohen and H. P. Flatt, "Numerical Solution of Quasi-Linear Equations," in *Codes for Reactor Computations,"* Intl. Atomic Energy Agency, Vienna (1961).

29. W. F. Walters and R. D. O'Dell, "Nodal Methods for Discrete-Ordinates Transport Problems in (x,y) Geometry," *Proc. Amer. NucL Soc. Top. Meeting on Advances in Mathematical Methods for the Solution of Engineering Problems,* Vol. I, pp. 115-129, Munich, FRG (April 1981).

30. W. F. Walters, "Recent Developments in Nodal and Characteristic Methods in Transport Theory," *Trans. Am. Nucl. Soc.* **43**, 611 (November 1982).

31. W. F. Walters, "Augmented Weighted Diamond Form of the Linear Nodal Scheme for Cartesian Coordinate Systems," *Proc. International Mtg. on Advances in Nuclear Engineering Computation Methods,* Knoxville, TN (April 1985).

32. R. L. Childs and W. A. Rhoades, "The Linear Nodal Method for Shielding Applications," *Trans. Am. Nucl. Soc.* **46**, 657-658 (June 1984).

33. R. L. Childs and W. A. Rhoades, "The Extension of the Linear Nodal Method to Large Concrete Buildings," *Trans. Am. Nucl. Soc.* **50**, 476-477 (November 1985).

34. R. E. Alcouffe and E. W. Larsen, "A Review of Characteristic Methods Used to Solve the Linear Transport Equation," *Proc. Amer. Nucl. Soc. Top. Meeting on Advances in Mathematical Methods for the Solution of Engineering Problems,* Vol. I, pp. 3-16, Munich, FRG (April 1981).

35. E. W. Larson and R. E. Alcouffe, "The Linear Characteristic Method for Spatially Discretizing the Discrete Ordinates Equations in (X,Y) Geometry," *Proc. Amer. Nucl. Soc. Top. Meeting on Advances in Mathematical Methods for the Solution of Engineering Problems,* Vol. I, pp. 99-113, Munich, FRG (April 1981).

36. K. P. Beerink and J. J. Dorning, "The Relationship Between the Nodal and Moments Characteristic Transport Methods and Their Errors," *Proc. Am. Nucl. Soc. Conf. on Advances In Reactor Computations,* pp. 737-756, Salt Lake City, Utah (March 1983).

37. 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).

38. C. 0. Slater, S. N. Cramer, and D. T. Ingersoll, "Analysis of the ORNL/TSF GCFR Grid-Plate Shield Design Confirmation Experiment," ORNL-5551 (August 1979).

39. 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 1: Fully Symmetric Quadratures and the DOQDP Computer Code, Part II: Common Quadrature Library," ORNL/TM-6023 (September 1977).

40. M. B. Emmett and W. A. Rhoades, "Monte Carlo Verification of DOT Cylindrical Boundary Condition," Oak Ridge National Laboratory, Oak Ridge TN (to be published).

41. C. J. Slavik, C. R. Lubitz, and J. T. Reynolds, "CLEM - A Legendre Polynomial Fitting Program With a Non-Negativity Constraint," KAPL-M-6923 (May 1968).

42. 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).

43. T. B. Fowler, D. R. Vondy, and G. W. Cunningham, "Nuclear Reactor Core Analysis Code: Citation,"ORNL/TM-2496, Rev. 2 (July 1969).

44. 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).

45. H. J. Kopp, "Synthetic Method Solution of the Transport Equation," *Nucl. Sci. Eng.* **17**, 65-74 (1963).

46. E. M. Gelbard and L. A. Hageman, "The Synthetic Method as Applied to the $S_n$ Equations," *Nucl. Sci. Eng.* **37**, 288-296 (1969).

47. W. A. Rhoades, "Improvements in Discrete Ordinates Acceleration," *Trans. Am. Nucl. Soc.* **39**, 753 (November 1981).

48. W. H. Reed, "The Effectiveness of Acceleration Techniques for Iterative Methods in Transport Theory," *Nucl. Sci. Eng.* **45**, 245-254 (1971).

49. 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).

50. R. W. Alcouffe, "Diffusion Synthetic Acceleration Methods for the Diamond-Differenced Discrete Ordinates Equations," *Nucl. Sci. Eng.* **64**, 344-355 (1977).

51. 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).

52. J. Aull, "The Stability of the Mesh-Cornered Synthetic Method of Diffusion Acceleration of the DOT-IV Transport Code," ORNL/TM-7097 (October 1979).

53. 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).

54. E. W. Larsen, "Diffusion-Synthetic Acceleration Methods for the Discrete Ordinates Equations," *Proc. Am. Nucl. Soc. Conf. on Advances In Reactor Computations,* pp. 705-724, Salt Lake City, Utah (March 1983).

55. E. M. Gelbard, D. R. McCoy, and E. W. Larsen, "Finite Difference Effects in the Synthetic Acceleration Method," *Trans. Am. Nucl. Soc.* **39,** 462 (November 1981).

56. E. W. Larsen and D. R. McCoy, "An Unconditionally Stable Acceleration Method for Arbitrary Weighted Differencing Schemes," *Trans. Am. Nucl. Soc.* **39,** 469 (November 1981).

57. E. W. Larsen, "Unconditionally Stable Diffusion-Synthetic Acceleration Methods for the Slab Geometry Discrete Ordinates Equations. Part I: Theory," *Nucl. Sci. Eng.* **82,** 47 (1982).

58. E. M. Gelbard, C. H. Adams, D. R. McCoy, and E. W. Larsen, "Acceleration of Transport Eigenvalue Problems," *Trans. Am. Nucl. Soc.* **41,** 309 (June 1982).

59. E. M. Gelbard and H. Khalil, "Alternative Differencing Technique for the Synthetic Method," *Trans. Am. Nucl. Soc.* **45,** 322 (October 1983).

60. H. Khalil, "A Nodal Diffusion Technique for Synthetic Acceleration of Nodal $S_n$ Calculations," *Nucl. Sci. Eng.* **90,** 263 (1985).

61. W. F. Miller, Jr., "Generalized Rebalance: A Common Framework for Transport Acceleration Methods," *Nucl. Sci. Eng.* **65,** 226 (1978).

62. J. E. Morel, "A Synthetic Acceleration Method For Discrete Ordinates Calculations with Highly Anisotropic Scattering," *Nucl. Sci. Eng.* **82,** 34 (1982).

63. ANS Standard "Recommended Programming Practices to Facilitate the Interchange of Digital Computer Programs," prepared by Subcommittee 10, ANS Standards Committee (April 1971).

64. "FORTRAN," Am. Natl. Standard ANSI X3.9-1966 (March 1966).

65. B. M. Carmichael, "Standard Interface Files and Procedures for Reactor Physics Codes, Version III," LA-5486-MS (February 1974).

66. R. Douglas O'Dell, "Standard Interface Files and Procedures for Reactor Physics Codes, Version IV," LA-6941-MS (September 1977).

67. "FORTRAN 77," Am. Natl. Standard ANSI X3.9-1978 (1978).

68. M. B. Emmett, "The MORSE Monte Carlo Radiation Transport Code System," ORNL-4972 (February 1975).

69. 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).

## 8. BIBLIOGRAPHY OF RELATED OAK RIDGE PUBLICATIONS


1. F. R. Mynatt, "A User's Manual for DOT," K-1694, unpublished.

   This originated as a rough draft of a Union Carbide document. Although it bore no date, it was distributed informally in or about 1968. It sufficed for both DOT and DOT II users.

2. 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).

   This described the space-dependent scaling introduced in DOT II.

3. W. W. Engle, Jr., F. R. Mynatt, and R. S. Booth, "One-Dimensional Time-Dependent Discrete Ordinates," *Trans. Am. Nucl. Soc.* **12**, 400-401 (June 1969).

   This was the first discussion of the weighted difference flux model used in DOT III.

4. F. R. Mynatt, F. J. Muckenthaler, and P. N. Stevens, "Development of a 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 behind DOT and extensive comparison with experiments.

5. F. R. Mynatt *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. It was also used for DOT 3.5, accompanied by a few pages of supplementary notes.

6. W. A. Rhoades, "The DOT IV Variable Mesh Discrete Ordinates Transport Code," *Proc. 5th Intl. Conf. Reactor Shielding,* pp. 724-730, Science Press (April 1977).

   This described general features of DOT IV and showed important advantages of the variable mesh features in solving streaming problems.

7. J. P. Jenal *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. 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).

This discussed the failure of weighted difference to provide accurate $k_{eff}$ calculations, although it converged faster. The "θ-weighted" model used in DOT IV was introduced. The new model combined rapid convergence with accuracy in the demonstration problem.

9. 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).

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. W. A. Rhoades *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. 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).

This compared DOT and TWOTRAN in speed, construction, and problem-solving capability. Although TWOTRAN lacked 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 until a later study, described below. DOT generally required fewer iterations to converge, due to the use of weighted difference.

12. 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).

This described a "Consistent Diffusion Acceleration" that brought compatibility with all of the difference schemes used in DOT IV and improved convergence on difficult problems.

13. 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).

This discussed stabilized conventional space rebalance (CRB), consistent diffusion acceleration (CDA), and an experimental "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 best, unless the problem contained large internal voids.

14. 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).

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 results into reasonably accurate answers.

15. J. E. Aull, "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) and ORNL/TM-7404 (July 1980).

A thorough discussion of consistent diffusion acceleration as applied to fixed-source transport problems was given in this thesis study.

16. J. E. Aull, "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. E. T. Tomlinson, 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. W. A. Rhoades, "Improvements in Discrete-Ordinates Acceleration," *Trans. Am. Nucl. Soc.* **39**, 753-755 (December 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. W. A. Rhoades and M. B. Emmett, "DOS: The Discrete Ordinates System," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL/TM-8362 (May 1982).

    The DOS system includes codes commonly used in conjunction with DOT: BNDRYS, GIP, and RTFLUM, together with a driver capable of executing several codes in sequence within a single job. The driver functioned only on IBM mainframe computers at that time, but it has since been made operable on UNIX systems as well.

20. W. A. Rhoades and R. L. Childs, "An Updated Version of the DOT 4 One- and Two-Dimensional Neutron/Photon Transport Code," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-5851 (July 1982).

    This described an updated version with new acceleration methods, improved programming, and adaptability to vector computers. Reviews of the theoretical basis and procedures were much more thorough than any previous ORNL document on the subject.

21. W. A. Rhoades, "DOT 4," *Proc. Am. Nucl. Soc. Advances in Reactor Computations,* pp. 197-199 (Salt Lake City, March 1983).

    The proceedings contained an up-to-date code abstract.

22. R. L. Childs and W. A. Rhoades, "The Linear Nodal Method For Shielding Applications," *Trans. Am. Nucl. Soc.* **46**, 657-658 (June 1984).

    This was the first two-dimensional Oak Ridge application of the linear nodal method as pioneered by Walters and O'Dell. The method proved fruitful for deep-penetration problems, but it required additional constraints to control behavior in extreme circumstances.

23. W. A. Rhoades, R. L. Childs, M. B. Emmett, and S. N. Cramer, "Application of the Three-Dimensional Oak Ridge Transport Code," *Proc. Am. Nucl. Soc. Topical Meeting on Reactor Physics and Shielding*," pp. 225-238 (Chicago, September 1984).

    In this first publication concerning TORT, the ability of the code to calculate penetration into concrete structures was demonstrated. A graphics package provided extensive displays. Comparison with Monte Carlo was made.

24. W. A. Rhoades, R. L. Childs, D. T. Ingersoll, and F. J. Muckenthaler, "Analysis of the TORT Validation Experiment," *Trans. Am. Nucl. Soc.* **50**, 473-475 (November 1985).

As a follow-on to the Chicago paper, a versatile concrete building was actually erected and tested. Selected comparisons with TORT calculations showed the extent of agreement.

25. R. L. Childs and W. A. Rhoades, "The Extension of the Linear Nodal Method to Large Concrete Buildings," *Trans. Am. Nucl. Soc.* **50**, 476-477 (November 1985).

    The linear nodal method was extended to three-dimensional concrete building applications. Additional controls were required to limit behavior. With these, results for large mesh cells were much superior to weighted difference.

26. J. L. Thompson, M. B. Emmett, W. A. Rhoades, and H. L. Dodds, Jr., "A Method For Coupling Two-Dimensional to Three-Dimensional Discrete Ordinates Calculations," *Trans. Am. Nucl. Soc.* **50**, 475-476 (November 1985).

    A method was demonstrated for synthesis of a three-dimensional boundary source for TORT from a two-dimensional flux file.

27. F. J. Muckenthaler, L. B. Holland, J. L. Hull, and J. J. Manning, "Verification Experiment of the Three-Dimensional Oak Ridge Transport Code (TORT)," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL/TM-9528 (December 1985).

    A concrete building was constructed and irradiated to provide a benchmark comparison for TORT. Numerous measurements were made of both neutron and gamma flux in a total of four building configurations.

28. W. A. Rhoades and R. L. Childs, "The TORT Three-Dimensional Discrete Ordinates Neutron/Photon Transport Code," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-6268 (April 1987).

    The DOT 4 methodology was extended to three dimensions without difficulty. The document contains a thorough review of theory, programming practices, and user's information. Principal application had been on vector computers such as Cray, but the code had also been used on IBM computers.

29. R. L. Childs and W. A. Rhoades, "The Linear Characteristic Method in XYZ Geometry," Trans. Am. Nucl. Soc. **55**, 352-354 (November 1987).

    The linear characteristic method was tested in a three-dimensional discrete ordinates calculation of penetration into an experimental concrete building. The new method was found to give lower error than weighted difference or linear nodal methods for a given mesh. Appropriate restraints on the expansion of boundary fluxes were discussed.

30. W. A. Rhoades and R. L. Childs, "$S_n$ Transport Calculations on Vector and Parallel Processors," Trans. Am. Nucl. Soc. **55**, 320-321 (November 1987).

The vectorization of the DOT two-dimensional discrete ordinates code was discussed. Different techniques were required for each section of the code. With the help of a limited amount of assembly language, the overall Cray speed was boosted to 10 times the IBM 3033 speed. Some of the changes also improved the IBM speed.

31. W. A. Rhoades and R. L. Childs, "The DORT Two-Dimensional Discrete Ordinates Transport Code," Nucl. Sci. & Engr. **99**, 1, 88-89 (May 1988).

This abstract described a version of DORT, the successor to DOT, operable on Cray-CTSS, Cray-COS, and IBM OS/MVS systems.

32. R. L. Childs, W. A. Rhoades, and L. R. Williams, "Three-Dimensional Calculations of Neutron Streaming in the Beam Tubes of the ORNL HFIR Reactor," Proc. 7th Internat. Conf. on Radiation Shielding, Bournemouth, UK (September 1988).

The TORT three-dimensional discrete ordinates transport code was used to calculate fluence data in and around beam tubes of the HFIR. It was a significant demonstration of the use of TORT inside a reactor vessel. After normalizing to available experimental data, the calculation provided spatial and spectral information not otherwise available. Although ray-streaming effects are always of concern when using discrete ordinates in a void, the results of these calculations were satisfactory.

33. W. A. Rhoades, R. L. Childs, and D. T. Ingersoll, "New Dose-Mortality Data Based on 3-D Radiation Shielding Calculations for Concrete Buildings at Nagasaki," Proc. 7th Internat. Conf. on Radiation Shielding, Bournemouth, UK (September 1988).

A summary paper described the methods, verification, and results from a state-of-the-art calculation of radiation to personnel inside two concrete buildings at Nagasaki. This is the application for which TORT was originally constructed.

34. W. A. Rhoades, R. L. Childs, and D. T. Ingersoll, "Radiation Exposure Inside Reinforced Concrete Buildings at Nagasaki," Oak Ridge National Laboratory, Oak Ridge, TN, ORNL/TM-10999 (December 1988).

A detailed description of methodology, verification, data, and results of the state-of-the-art calculation of dose to personnel inside two concrete buildings at Nagasaki was given. Several options are compared with each other and with a Monte Carlo calculation with respect to accuracy. A detailed error analysis was performed, and the random uncertainty was compared with the random scatter of actual observations. A companion effort performed elsewhere used

these data to obtain a new value of the LD50, a parameter of international importance in radiation protection matters.

35. W. A. Rhoades and R. E. Flanery, "3-D Discrete Ordinates Calculatons with Parallel-Vector Processors," Proc. Am. Nucl. Soc. Topical Meeting on Advances in Nuclear Engineering Computation and Radiation Shielding, pp. 69:1-69:11 (Santa Fe, April 1989).

    The CPU-intensive linear nodal method was reorganized to allow two or more Cray CPU's to work concurrently on a problem. The value of the method was demonstrated, and a significant cost reduction was observed. The long time required to attach additional CPU's made the use of more than two processors impractical in this study.

36. W. A. Rhoades and R. L. Childs, "TORT: A Three-Dimensional Discrete Ordinates Neutron/Photon Transport Code," Nucl. Sci. & Engr. 107, **4**, pp. 397-398 (April 1991).

    This abstract gave a quick overview of the first version of the code released for general use. This version was operable only with the Cray UNICOS operating system.

37. W. A. Rhoades and R. L. Childs, "TORT -- Three-Dimensional Discrete Ordinates Neutron/Photon Transport Code," Proceedings of the Am. Nucl. Soc. Topical Mtg.: Advances in Mathematics, Computations, and Reactor Physics (Pittsburgh PA, April 1991).

    This abstract gave a quick overview of an updated version operable on Cray -UNICOS, IBM-AIX, and other UNIX systems.

38. W. A. Rhoades, R. L. Childs, and D. T. Ingersoll, "Radiation Exposure Inside Reinforced Concrete Buildings at Nagasaki," Health Physics **63**, 5, 510-521 (November 1992).

    This was the formal publication of the material detailed in ORNL/TM-10999. Some additional data had been added, and the classification of cases had been revised.

39. R. L. Childs and W. A. Rhoades, "Theoretical Basis of the Linear Nodal and Linear Characteristic Methods in the TORT Computer Code," ORNL/TM-12246 (January 1993).

    This detailed derivation used special techniques to control negative flux generation and numerical flux distortions that are likely to occur in shielding applications.

40. W. A. Rhoades and R. L. Childs, "Applications of TORT Three Dimensional Neutron/Photon Transport Calculations," Eighth ASTM-EURATOM Symposium on Reactor Dosimetry (Vail, Colorado, August 1993).

    This paper summarized various applications of TORT.

41. W. A. Rhoades, "The TORSED Method for Construction of TORT Boundary Sources from External DORT Flux Files," ORNL/TM-12359 (August 1993).

TORSED allowed efficient coupling of a 2-dimensional DORT environment calculation to a internally-contained section to be calculated by TORT.

42. W. A. Rhoades, " The TORSED and TORSET Codes for Coupling Three-Dimensional TORT Calculations," Proceedings, 8th International Conference on Radiation Shielding, Am. Nucl. Soc. No. 700201, ISBN 0-89448-191-6 pp. 551-557 (Arlington TX, March 1994).

New procedures had been added to TORSED in this paper, and it was joined by TORSET, which couples one three-dimensional TORT calculation to another. The second problem is allowed to lie partly outside the first.

# APPENDICES

# APPENDIX A. FIDO INPUT

The FIDO input method is designed to facilitate the entering or modifying of large data arrays. Special advantage is taken of repetition or symmetry wherever possible. The FIDO system was patterned after the input method used with the FLOCO coding system at Los Alamos. It was originally developed at Atomics International for the DTF-II[69] 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 that 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. Arrays are grouped into *blocks*, each representing a call to the FIDO package. A block may contain one, many, or no arrays. 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. An array can be repeated within the same block. For example, an array can be filled with 0 using a special option, and then a few scattered locations can be changed by reading in a new set of data for that array. An array can be omitted if no entries are required. If none of the arrays in a block are required, but the block is required, the block delimiter satisfies the input requirement.

Three major types of input are available: free-field input, fixed-field input, and user-field input. Each field includes up to three subfields. Each array is identified by an *array originator field* having two subfields:

Subfield 1: An integer *array identifier* signals the identity of the array to follow.

Subfield 2: An *array type indicator*, used as follows:

| | |
|---|---|
| "$$" | free-field integer array |
| "**" | free-field real array |
| "$" | fixed-field integer array |
| "*" | fixed-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. It should be noted that capitals are used to represent character fields here for the sake of clarity. The actual entries, however, must follow the requirement of the computer, which may very likely demand lower-case entries.

In entering data, it is convenient to think of a *pointer* that is under control of the user, and that specifies the location in the array into which the next data entry is to be placed. The pointer is always positioned at the first array location by 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 *subfields*:

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 separate subfield 2 from subfield 3, however. All entries following a "/" in a given record are ignored. Do not use the TAB character to separate fields!

*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 into data array 1
2**   1. 1.1  2  /enter reals 1.0, 1.1, 2.0 into data array 2
```

An exponent field may be added, with or without the "e" identifier. No imbedded blanks within the subfield are allowed. The subfield must have no more than 9 columns, including the decimal, but 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
AND NOT   3**  0.0e6               /produces wrong results
```

*Multiple data entries* use $N_2$ and (except the "f" operator) $N_1$, together with $N_3$ to enter several items with a single field. The type of operation is indicated by $N_2$.

"$N_1$r $N_3$" indicates that data entry $N_3$ is to be repeated $N_1$, times. The pointer is advanced by $N_1$.

"$N_1$i $N_3$" 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 value 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 defining a spatial mesh. In integer arrays, interpolated values are rounded to the nearest integer.

"$N_1$l $N_3$" 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 obviously limited to positive real numbers.

"f $N_3$" 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$$  1 2 3 4 6 6 8 8
2$$ 2i1 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 $N_1$, $N_2$,and $N_3$ are used, except that, if $N_1$ is omitted, it is taken to be 1.

"$N_1$q $N_3$" 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 in specifying matrices with repeated or symmetrical columns or rows.

"$N_1$g $N_3$" has the same effect as "q", except that the sign of each entry of the sequence is changed each time it is entered.

"$N_1$n $N_3$" has the same effect as "q", except that the order of the sequence is reversed each time it is entered.

"$N_1$m $N_3$" has the same effect as "q", except that both the sign and the order of the sequence are reversed each time it is entered.

Options q, g, and m are valuable in entering directional quadrature sets. As examples:

```
81**   1 2  3  1q3      /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 $N_2$.

"$N_1$z" sets the next $N_1$, entries to 0. The pointer is advanced by $N_1$. As an example:

```
1$$ 3z 1 2          /0, 0, 0, 1, 2
```

*Pointer-movement data entries* move the pointer without changing the data array. Subfields $N_1$ and $N_3$ may or may not may be required.

"$N_1$s" indicates that the pointer is to skip forward over $N_1$ positions, leaving those array positions unchanged.

"$N_1$b" moves the pointer backward $N_1$ positions.

"a $N_3$" moves the pointer to the position to $N_3$.

"e" skips over the remainder of the array. The array length criterion is satisfied by an "e", unless too many entries have been specified previously. No more entries to an 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:

```
1$$ 1 2 3 4 5 6 e
/E terminates the array, leaving

/items 7 and 8 unchanged
1$$ 2 1s 4 a7 7 8
/now we have 2,2,4,4,5,6,7,8
1$$ 2i 1 4 1b 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 record 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. Any entries following a block termination field in a record are ignored.

The reading of data to an array is terminated when a new array originator field is supplied, or when a block termination field is encountered. If the final pointer location is not correct, an error edit is given, and a flag that will later abort execution of the problem is set. FIDO then continues with the next array if an array originator was read. The new array originator need not begin a new record. If a block termination field was read, FIDO returns control to the calling program. For example:

```
1$$  1 2 3      2** f0 t
```

"u" as an array type indicator specifies user-field input, allowing the user to specify the input format. The FORTRAN format to be used must be supplied in columns 1-72 of the next record. The format must be enclosed by 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 ensure that the 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 most recent "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 described in the DOT IV document.[25] It has been almost entirely replaced by the free-field format, and so its description is not repeated here.

# APPENDIX B. INTERFACE FILE SPECIFICATIONS

Specifications for the sequential files normally used to carry information into and out of TORT are given in Table B.1.

In addition, some direct-access files that are normally scratch files can be saved from one problem and used in another. These formats are given in Table B.2. The user should note that the direct-access files do not have indentification of the type that heads the sequential files, so that the burden of assuring that the correct file is in use is left entirely to the user.

**Sequential formats include:**

dirflx cell-average directional flux

flxmom cell-average flux moments and boundary directional flux

gip macroscopic cross sections

varbnd boundary directional source

varmap zone map data

varscl cell-average scalar flux, zone map data and convergence data

**Direct-access formats include:**

torscrf cell-average flux/source moments

torscrs boundary directional fluxes/sources

Table B.1. Format Descriptions For Sequential Interface Files

```
------------------------------------------------------------------------
- name:      dirflx
-
- date:      01 jun 95
-
- purpose:   cell-average directional flux and interpretive data
-
- notes:     order of energy groups is by decreasing energy --
-           neutrons, then photons.
-
-  i is the first -dimension index, i=1,,,ims(j,k)
-  j is the second-dimension index, j=1,,,jms(k)
-  k is the third -dimension index, k=1,,,km
-  m is the overall direction index, m=1,,,mset(ig)
-  mx is the quadrant direction index, mx=1,,,mdim
-  ig is the energy group index, ig=1,igm
-
-  mult=1 if word length is 8 bytes; mult=2 if 4 bytes.
-
-  the "output patch" refers to a compact region of the space mesh in
-    which the directional flux is to be output.
-
-  flux records for a group are not present if no iterations are done on
-  that group.  the user must provide for this possibility.
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- file structure:
-
-      record type                          present if
-      -----------                          ----------
-      file identification                    always
-      file label                             always
-      indexing arrays                        always
-      real arrays                            always
-
-  ............repeat for ig=1,igm
-  .
-  . ..........repeat all k=km,1,-1
-  . . ........repeat all j=jms(k),1,-1
-  . . . in, downward  directional flux     always
-  . . ........end j loop
-  . . ........repeat all j=1,jms(k),1
-  . . . out, downward directional flux     always
-  . . ........end j loop
-  . ..........end k loop
-  .
-  . ..........repeat all k=1,km
-  . . ........repeat all j=jms(k),1,-1
-  . . . in, upward    directional flux     always
-  . . ........end j loop
-  . . ........repeat all j=1,jms(k)
-  . . . out, upward  directional flux      always
-  . . ........end j loop
```

```
-  . ..........end k loop
-  .
-  ............end ig loop
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- file identification:
-
-     hname,(huse(i),i=1,2),ivers
-
-     number of words= 4*mult
-
-     hname              file name                       - (a8)
-     huse(i)            user file identification        - (a8)
-     ivers              file version number             - (a8)
------------------------------------------------------------------------


------------------------------------------------------------------------
- file label:
-
-     date,user,charge,case,time,(titl(i),i=1,9)
-
-     number of words= 14*mult
-
-     date               as provided by timer option 4 - (a8)
-     user               as provided by timer option 5 - (a8)
-     charge             as provided by timer option 6 - (a8)
-     case               as provided by timer option 7 - (a8)
-     time               as provided by timer option 8 - (a8)
-     titl(i)            title provided by user        - (a8)
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- integer parameters:
-
-     igm,im,jm,km,mmdnup, mm,mmsmsm,msm,ipch1,ipch2
-     jpch1,jpch2,kpch1,kpch2, (idum(n),n=1,11)
-
-     number of words= 25
-
-     igm                number of energy groups
-     im                 maximum number of cells in any row
-     jm                 maximum number of rows in any plane
-     km                 number of planes
-     mmdnup             max. no. of down/up directions in any m-set
-
-     mm                 max. no. of directions in any m-set
-     mmsmsm             sum of no. of directions over m-set
-     msm                number of m-sets
-     ipch1              first i in output patch, first i-set
-     ipch2              last  i in output patch, first i-set
```

```
-
-     jpch1              first j in output patch, first j-set
-     jpch2              last  j in output patch, first j-set
-     kpch1              first j in output patch, first k-set
-     kpch2              last  j in output patch, first k-set
-     idum               array set to 0
-
-------------------------------------------------------------------------
-------------------------------------------------------------------------
- indexing arrays:
-     (mmbms(ms),ms=1,msm), (mset(ig),ig=1,igm)
-   , (mmdn(ms),ms=1,msm), (mmdu(ms),ms=1,msm)
-
-     number of words= 3*msm+igm
-
-     mmbms              no. of directions by m-set
-     msets              no. of m-set by group
-     mmdn               no. of downward directions by m-set
-     mmdu               max. no. of down/upward directions by m-set
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- real arrays:
-
-     (  w(m,ms),m=1,mmbms(ms)),ms=1,msm)
-   , (emu(m,ms),m=1,mmbms(ms)),ms=1,msm)
-   , (xzi(m,ms),m=1,mmbms(ms)),ms=1,msm)
-   , (eta(m,ms),m=1,mmbms(ms)),ms=1,msm)
-   , (pchbn(l),l=1,6)
-
-     number of words = 4*mmsmsm+6
-
-     w                  direction weight
-     emu                direction cosine with x axis
-     xzi                direction cosine with y axis
-     eta                direction cosine with z axis
-     pchbn              left/right,inside,outside,bottom,top patch bound
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- directional flux:
-
-     ((dirf(i,mx),i=1,ims(j,k)),mx=1,mdim)
-
-     number of words = ims*mdim
-
-     dirf               cell-average directional flux
-
------------------------------------------------------------------------

      end
```

```
------------------------------------------------------------------------
-
- name:      flxmom
-
- date:      9 february 1994
-
- purpose:   cell-average scalar data, cell average moment data,
-           and/or boundary directional data
-
- notes:     order of groups is by decreasing energy;
-           neutrons, then photons.
-
-  i is the first -dimension index.
-  j is the second-dimension index.
-  k is the third -dimension index.
-  zero is a dummy word.
-
-  mult = single precision word length indicator:
-         1 if word length.ge.6 bytes,
-         2 if word length.lt.6 bytes.
-
-  scalar response output uses ifmom=1, ifbnd=0, lm=0.
-
-  distributed source input uses ifmom=1, ifbnd=0, lm>0.
-
-  flux restart output uses ifmom=1, ifbnd=1, lm>0.
-  (at this time, dummy records are written for boundary flux.)
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- file structure:
-
-      record type                          present if
-      -----------                          ----------
-      file identification                  always
-      file label                           always
-      integer parameters                   always
-      integer arrays                       always
-      real parameters & arrays             always
-
-  ............do ig=1,igm                 if igm.gt.0
-  .
-  . ..........do nr=1,nresp               if nresp.gt.0
-  . .
-  . .   cell scalar data                  if ifmom.gt.0 .and. lm.eq.0
-  . .
-  . . .........do k=1,km
-  . . . .......do j=1,jm
-  . . . .  cell moment data               if ifmom.gt.0 .and. lm.gt.0
-  . . . .......enddo on j
-  . . .........enddo on k
-  . .
-  . .   i-boundary directional data       if ifbnd.gt.0
-  . .   j-boundary directional data       if ifbnd.gt.0
-  . .   k-boundary directional data       if ifbnd.gt.0
-  . .
-  . ..........enddo on nr
-  ............enddo on ig
```

```
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- file identification:
-
-     hname,(huse(i),i=1,2),ivers
-
-     number of words= 4*mult
-
-     hname              file name                       - (a6)
-     huse(i)            user identification             - (a6)
-     ivers              file version number             - (a6)
------------------------------------------------------------------------


------------------------------------------------------------------------
- file label:
-
-     date,user,charge,case,time,(titl(i),i=1,12)
-
-     number of words= 17*mult
-
-     date               as provided by timer option 4 - (a6)
-     user               as provided by timer option 5 - (a6)
-     charge             as provided by timer option 6 - (a6)
-     case               as provided by timer option 7 - (a6)
-     time               as provided by timer option 8 - (a6)
-     titl(i)            title provided by user        - (a6)
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- integer parameters:
-
-     ifmom,ifbnd,idim, igm,neut, im,jm,km,lm,mm, nresp,nconv
-    , njblk, (idum(n),n=1,12)
-
-     number of words= 25
-
-     ifmom              1 if moment or scalar records are present
-     ifbnd              1 if boundary records are present
-     idim               1/2/3 for 1-d, 2-d, or 3-d
-     igm                number of energy groups; 0 for sum over grps
-     neut               last neutron group
-                           (igm if all neutrons, 0 if all photons)
-     im                 number of i intervals
-     jm                 number of j intervals (1 if idim.lt.3)
-     km                 number of k intervals (1 if idim.eq.1)
-     lm                 length of moment expansion; 0 for scalar data
-     mm                 number of quadrature directions
-
-     nresp              number of responses
-     nconv              number of last energy group converged
-     njblk              number of space mesh blocks per group
-     idum(i)            array set to 0
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- integer arrays:
-
-     (iduma(i),i=1,10)
-
-     number of words= 10
```

```
-
-     iduma(i)          array set to 0
-
------------------------------------------------------------------------
------------------------------------------------------------------------
- real parameters & arrays:
-
-     (x(i),i=1,im+1),(y(j),j=1,jm+1),(z(i),i=1,km+1)
-   , (ener(ig),ig=1,igm),emin,eneut, (dampg(ig),ig=1,2*igm)
-
-     number of words = km+jm+im+3+3*igm+2
-
-     x(i)              i-interval boundaries
-     y(j)              j-interval boundaries (absent if idim.le.2)
-     z(k)              k-interval boundaries (absent if idim.eq.1)
-
-     ener(ig)          top energy boundary of group ig
-     emin              bottom energy boundary of group igm
-     eneut             bottom energy boundary of group neut
-                          (0 if neut=0)
-     dampg             restart acceleration damping data
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- cell scalar data:
-
-     (((flij(i,j,k),i=1,im),j=1,jm),k=1,km)
-
-     number of words = im*jm*km
-
-     flij              cell-average scalar data
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- cell moment data:
-
-     ((flum(i,l),i=1,im),l=1,lm)
-
-     number of words = im*lm
-
-     repeat above for jm*km records
-
-     flum              cell-average moment data
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- i-boundary directional data:
-
-     zero
-
-     number of words = 1
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- j-boundary directional data:
-
-     zero
-
```

```
-     number of words = 1
-
------------------------------------------------------------------------
------------------------------------------------------------------------
- k-boundary directional data:
-
-     zero
-
-     number of words = 1
-
------------------------------------------------------------------------

      end
```

```
------------------------------------------------------------------------
-
- name:     gip
-
- date:     20 jan 85
-
- purpose:  cross sections group-ordered for transport codes; this
-          format is compatible with anisn and dot requirements.
-
- notes:
-
-    groups are normally ordered from high neutron energy to low,
-  then from high photon energy to low, if any.  igm is the number of
-  groups.
-
-    in a forward (non-adjoint) calculation, scatters appearing
-  before ihs are upscatter; scatters appearing after ihs are
-  downscatter.  position ihp contains total upscatter, if any.
-
-    if upscatter data are not present, ihp position is missing
-  and ihs.eq.iht+1.
-
-     each component of a pl set is treated as a separate material
-  in this format.
-
-     transfers from groups .lt.1 or .gt.igm are 0.  positions
-  .le.iht are 0 for pl components other than 0'th.
-
-     for adjoint calculations, reverse the scattering; i.e. where
-  the format asks for scattering from ig-1 to ig, for example,
-  supply physical data for scattering from ig to ig-1.
-  next, reverse the ordering of the group data so that data for
-  the highest neutron energy is last, not first, etc.
-  if ihs-1.gt.iht, the position ihp has its same formal definition,
-  but the physical data it describes are different--it now describes
-  total upscatter into a group, in physical terms.
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- file structure:
-
-      record type                          present if
-      -----------                          ----------
-
-   ...........do ig=1,igm
-   .  cross section data                   always
-   ...........enddo
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- cross section data:
-
-     ((sig(ih,mt),ih=1,ihp),mt=1,mtm)
-
-     number of words = ihp*mtm
-
-     sig(ih,mt)         cross section data by table position, then by
-                        nuclide.  table positions contain - -
-
-          1 to iht-5  arbitrary data, specified by user, or absent
-               iht-4  fission yield fraction (recommended)
-               iht-3  fission cross section  (recommended)
```

```
-               iht-2  absorption cross section
-               iht-1  neutron production cross section
-               iht    total cross section
-               iht+1  scatter from group ig+(ihs-1-iht) to ig
-                   .
-                   .
-               ihs-1  scatter from group ig+1 to ig
-               ihs    self scatter for group ig
-               ihs+1  scatter from group ig-1 to ig
-                   .
-                   .
-               ihm    scatter from group ig-(ihm-ihs) to ig
-               ihp    total scatter from group g to groups 1,...,g-1
-------------------------------------------------------------------------
```

```
------------------------------------------------------------------------
-
- name:      varbnd
-
- date:      03 march 1993
-
- purpose:   boundry source and associated interpretation data
-
- notes:     order of energy groups is by decreasing energy --
-           neutrons, then photons.
-
-  i is the first -dimension index.
-  j is the second-dimension index.
-  k is the third -dimension index.
-  m is the direction index.
-  zero is a word set to zero used in padding to full length.
-
-  mult=1 if word length is 8 bytes; mult=2 if 4 bytes.
-
-  when im.gt.0, the mesh is a regular (continuous) mesh with im cells
-  in each row and jm rows in each plane.  ism=jsm=ksm=1.  ims=im.
-  jms=jm.  ima=im.
-
-  when im.lt.0, the mesh is discontinuous.  each plane contains
-  jms rows, where jms=jmbjs(jset(k)).  each row contains ims cells,
-  where ims=imbis(iset(j'k)).  (j'k) denotes j + sum of jms(kk)
-  over kk=1,...,k-1.  ima=iabs(im).  ism is the number of i-sets.
-  jms is the number of j-sets.
-
-  when mm.gt.0, mm directions are used in the directional quadrature of
-  flux in each energy group.  msm=1.  mms=mma.
-
-  when mm.lt.0, the number of directions in the directional quadrature
-  varies by group.  mms=mmbms(mset(ig)) is the number of directions
-  used in group ig.  mma=iabs(mm).  msm is the number of direction
-  sets.
-
-  mmsdu(ig) is the larger of the number of downward or upward
-  directions for the m-set used in ig.  mmdnup is the largest mmsdu
-  for any ig.
-
-  for use by tort, ifbfxi=ifbfxj=ifbfxk=0.
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- file structure:
-
-       record type                          present if
-       -----------                          ----------
-       file identification                  always
-       file label                           always
-       integer parameters                   always
-       direction indexing arrays            always
-       space indexing arrays                always
-       real arrays                          always
-
-  ............do ig=1,igm
-  .
-  . ..........do k=km,1,-1
-  . .    i-boundary directional data     if ifbfxi.eq.0
-  . .    j-boundary directional data     if ifbfxj.eq.0
-  . ..........enddo on k
-  .
-  . ..........do j=1,jm
```

```
-  . .   k-boundary source, top          if ifbfxk.eq.0
-  . .........enddo on j
-  .
-  . .........do j=1,jm
-  . .   k-boundary source, bottom       if ifbfxk.eq.0
-  . .........enddo on j
-  .
-  ...........enddo on ig
-
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- file identification:
-
-     hname,(huse(i),i=1,2),ivers
-
-     number of words= 4*mult
-
-     hname              file name                        - (a8)
-     huse(i)            user file identification         - (a8)
-     ivers              file version number              - (a8)
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- file label:
-
-     date,user,charge,case,time,(titl(i),i=1,9)
-
-     number of words= 14*mult
-
-     date               as provided by timer option 4 - (a8)
-     user               as provided by timer option 5 - (a8)
-     charge             as provided by timer option 6 - (a8)
-     case               as provided by timer option 7 - (a8)
-     time               as provided by timer option 8 - (a8)
-     titl(i)            title provided by user        - (a8)
-
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- integer parameters:
-
-     igm,im,jm,km,mm, mmdnup,ism,jsm,imsism,jmsjsm, jmskm,msm,mmsmsm
-    , ifbfxi,ifbfxj, ifbfxk,(idum(n),n=1,9)
-
-     number of words= 25
-
-     igm                number of energy groups
-     im                 +/- maximum number of cells in any i-set
-     jm                 maximum number of rows in any j-set
-     km                 number of planes
-     mm                 +/- maximum number of directions in any m-set
-
-     mmdnup             maximum number of directions down or up
-                          in any m-set
-     ism                number of i-sets
-     jsm                number of j-sets
-     imsism             sum of ims over is
-     jmsjsm             sum of jms over js
-
-     jmskm              sum of jms over km
-     mmsmsm             sum of mmbms over ms
-     msm                number of m-sets
```

```
-     ifbfxi              .eq.0 if i-boundary flux is included, else 1
-     ifbfxj              .eq.0 if j-boundary flux is included, else 1
-
-     ifbfxk              .eq.0 if k-boundary flux is included, else 1
-     idum                array set to 0
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- direction indexing arrays:
-     (mmbms(ms),ms=1,msm),(mset(ig),ig=1,igm)
-
-     number of words= msm+igm
-
-     mmbms               number of directions in m-set ms
-     mset                m-set assigned to energy group ig
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- space indexing arrays:
-     (imbis(is),is=1,ism), (jmbjs(js),js=1,jsm)
-   , ((iset(j'k),j=1,jms),k=1,km), (jset(k),k=1,km)
-
-     number of words= ism+jsm+jmskm+km
-
-     imbis               number of cells in i-set is
-     jmbjs               number of cells in j-set js
-     iset                i-set assigned to row j in plane k
-     jset                j-set assigned to plane k
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- real arrays:
-
-     ((x(i,is),i=1,ims+1),is=1,ism), ((y(j,js),j=1,jms+1),js=1,jsm)
-   , (z(k),k=1,km+1), (ener(ig),ig=1,igm),emin,eneut, (dumrl(i),i=1,8)
-
-     number of words = imsism+ism+jmsjsm+jsm+km+1+igm+2+8
-
-     x                   i-interval boundaries by i-set
-     y                   j-interval boundaries by j-set
-     z                   k-interval boundaries
-
-     ener                top energy boundary of group ig
-     emin                bottom energy boundary of group igm
-     eneut               bottom energy boundary of group neut
-                            (0 if neut=0)
-     dumrl               array set to 0.
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- i-boundary source:
-
-     ((fio(m,j),m=1,mms),j=1,jms), (zero,l=1+mms*jms,mms*jm)
-
-     number of words = mms*jm
-
-     fio    i-boundary directional source
-
```

```
-----------------------------------------------------------------------


-----------------------------------------------------------------------
- j-boundary source:
-
-     ((fjo(m,i),m=1,mms),i=1,ims), (zero,l=1+mms*ims,mms*ima)
-
-     number of words = mms*ima
-
-     fjo    j-boundary directional source
-
-----------------------------------------------------------------------


-----------------------------------------------------------------------
- k-boundary source (top or bottom):
-
-     ((fko(m,i),m=1,mmsdu),i=1,ims), (zero,l=1+mmsdu*ims,mmsdu*ima)
-
-     number of words = mmsdu*ima
-
-     fko    k-boundary directional source, downward or upward
-            (for j.gt.jms, fko is filled with zero.)
-
-----------------------------------------------------------------------

      end
```

```
------------------------------------------------------------------------
-
- name:     varmap
-
- date:     02 jun 93
-
- purpose:  material zone map
-
- notes:
-
-  i is the first -dimension index, i=1,,,ims(j,k)
-  j is the second-dimension index, j=1,,,jms(k)
-  k is the third -dimension index, k=1,,,km
-  ig is the energy group index, ig=1,igm
-
-  when im.gt.0, the mesh is a regular (continuous) mesh with im cells
-  in each row and jm rows in each plane.  ism=jsm=ksm=1.  ims=im.
-  jms=jm.
-
-  when im.lt.0, the mesh is discontinuous.  each plane contains
-  jms rows, where jms=jmbjs(jset(k)).  each row contains ims cells,
-  where ims=imbis(iset(j'k)).  (j'k) denotes j + sum of jms(kk)
-  over kk=1,...,k-1.
-
-  mult=1 if word length is 8 bytes; mult=2 if 4 bytes.
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- file structure:
-
-      record type                        present if
-      -----------                        ----------
-      file identification                always
-      file label                         always
-      integer parameters                 always
-
-  ............repeat all k=1,km
-  .     cell zone identification         always
-  ............end k loop
-
-      i-set by j and k                   always
-      length of each i-set               always
-      j-set by k                         always
-      length of each j-set               always
-
-  ............repeat all is=1,ism
-  .     i-boundaries                     always
-  ............end ism loop
-
-      j-boundaries by j-set              always
-      k-boundaries                       always
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- file identification:
-
-     hname,(huse(i),i=1,2),ivers
-
-     number of words= 4*mult
-
-     hname              file name                        - (a8)
-     huse(i)            user file identification         - (a8)
```

```
-     ivers              file version number            - (a8)
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- file label:
-
-     date,user,charge,case,time,(titl(i),i=1,9)
-
-     number of words= 14*mult
-
-     date               as provided by timer option 4 - (a8)
-     user               as provided by timer option 5 - (a8)
-     charge             as provided by timer option 6 - (a8)
-     case               as provided by timer option 7 - (a8)
-     time               as provided by timer option 8 - (a8)
-     titl(i)            title provided by user        - (a8)
-
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- integer parameters:
-
-     im,jm,km,imsjm,ism, jsm,(idum(n),n=1,19)
-
-     number of words= 25
-
-     im                 max. number of mesh cells in any row
-     jm                 max. number of rows in any plane
-     km                 number of planes
-     imsjm              number of mesh cells in largest plane
-     ism                number of i-sets
-
-     jsm                number of j-sets
-     imsism             number of mesh cells in a row, summed over row
-     jmsjsm             number of mesh rows per plane, summed over plane
-     idum               array set to 0
-
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- cell zone identification:
-
-     ((ijzn(i,j),i=1,ims),j=1,jms),(zero(i),i=1,ifill)
-
-     number of words = imsjm
-
-     ijzn               material zone by cell
-     zero               zero array required to pad length to imsjm
-     ifill              length of pad required (may be 0)
-
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- i-set by j and k:
-
-     ((iset(j,k),j=1,jms),k=1,km),(zeroa(i),i=1,ifilla)
-
```

```
-     number of words= jm*km
-
-     iset                number of i-set by j and k
-     zeroa               zero array required to pad length to jm*km
-     ifilla              length of pad required (may be 0)
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- length of each i-set:
-
-     (imbis(is),is=1,ism),(zerob(i),i=1,ifillb)
-
-     number of words= jm*km
-
-     imbis               length of i-set by i-set
-     zerob               zero array required to pad length to jm*km
-     ifillb              length of pad required (may be 0)
------------------------------------------------------------------------


------------------------------------------------------------------------
- j-set by k:
-
-     (jset(k),j=1,km)
-
-     number of words= km
-
-     jset                number of j-set by k
------------------------------------------------------------------------


------------------------------------------------------------------------
- length of each j-set:
-
-     (jmbjs(js),js=1,jsm),(zeroc(i),i=1,ifillc)
-
-     number of words= km
-
-     jmbjs               length of j-set by j-set
-     zeroc               zero array required to pad length to km
-     ifillc              length of pad required (may be 0)
------------------------------------------------------------------------


------------------------------------------------------------------------
- i-boundaries:
-
-    (rin(i,is),i=1,ims+1)
-
-    number of words = ims+1
-
-     rin                 i-boundaries by i-set
------------------------------------------------------------------------


------------------------------------------------------------------------
- j-boundaries by j-set
-
-    ((thin(j,js),i=1,jms+1),js=1,jsm)
-
-    number of words = sum of jms+1 over js
-
```

```
-     thin                  j-boundaries by j-set
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- k-boundaries:
-
-     (zin(k),k=1,km+1)
-
-     number of words = km+1
-
-     zin                   k-boundaries by k-set
-------------------------------------------------------------------------

      end
```

```
-------------------------------------------------------------------------
-
- name:     varscl
-
- date:     9 february 1994
-
- purpose:  scalar flux and associated interpretation data
-
- notes:    order of energy groups is by decreasing energy --
-          neutrons, then photons.
-
-  i is the first -dimension index.
-  j is the second-dimension index.
-  k is the third -dimension index.
-  ima=iabs(im).
-
-  mult=1 if word length is 8 bytes; mult=2 if 4 bytes.
-
-  when im.gt.0, the mesh is a regular (continuous) mesh with im cells
-  in each row and jm rows in each plane.  ism=jsm=ksm=1.  ims=im.
-  jms=jm.
-
-  when im.lt.0, the mesh is discontinuous.  each plane contains
-  jms rows, where jms=jmbjs(jset(k)).  each row contains ims cells,
-  where ims=imbis(iset(j'k)).  (j'k) denotes j + sum of jms(kk)
-  over kk=1,...,k-1.
-
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- file structure:
-
-      record type                       present if
-      -----------                       ----------
-      file identification               always
-      file label                        always
-      integer parameters                always
-      integer arrays                    always
-      indexing arrays                   always
-      real arrays                       always
-
-  ............repeat all k=1,km
-  .     cell zone identification        always
-  ............end k loop
-
-  ............repeat for ig=1,igm
-  . ..........repeat all k=1,km
-  . .   cell scalar flux                always
-  . ..........end k loop
-  ............end ig loop
-
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- file identification:
-
-     hname,(huse(i),i=1,2),ivers
-
-     number of words= 4*mult
-
-     hname              file name                      - (a8)
-     huse(i)            user file identification       - (a8)
-     ivers              file version number            - (a8)
-------------------------------------------------------------------------
```

```
------------------------------------------------------------------------
- file label:
-
-     date,user,charge,case,time,(titl(i),i=1,9)
-
-     number of words= 14*mult
-
-     date              as provided by timer option 4 - (a8)
-     user              as provided by timer option 5 - (a8)
-     charge            as provided by timer option 6 - (a8)
-     case              as provided by timer option 7 - (a8)
-     time              as provided by timer option 8 - (a8)
-     titl(i)           title provided by user        - (a8)
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- integer parameters:
-
-     igm,im,jm,km,izm, neut,ism,jsm,imsism,jmsjsm, jmskm,nconv
-   , njblk, (idum(n),n=1,12)
-
-     number of words= 25
-
-     igm               number of energy groups
-     im                maximum number of cells in any row
-     jm                maximum number of rows in any plane
-     km                number of planes
-     izm               number of material zones
-
-     neut              last neutron group
-                          (igm if all neutrons, 0 if all photons)
-     ism               number of isets
-     jsm               number of jsets
-     imsism            sum of ims over is
-     jmsjsm            sum of jms over js
-
-     jmskm             sum of jms over km
-     nconv             number of last group converged
-     njblk             number of space mesh blocks per group
-     idum              array set to 0
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- integer arrays:
-     (izcomp(iz),iz=1,izm),(iznrg(iz),iz=1,izm)
-
-     number of words= izm+izm
-
-     izcomp            material component by material zone
-     iznrg             edit region by material zone
-
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- indexing arrays:
-     (imbis(is),is=1,ism), (jmbjs(js),js=1,jsm)
-   , ((iset(j'k),j=1,jms),k=1,km), (jset(k),k=1,km)
-
```

```
-     number of words= ism+jsm+jmskm+km
-
-     imbis              number of cells in i-set is
-     jmbjs              number of cells in j-set js
-     iset               i-set assigned to row j in plane k
-     jset               j-set assigned to plane k
-
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- real arrays:
-
-     ((x(i,is),i=1,ims+1),is=1,ism), ((y(j,js),j=1,jms+1),js=1,jsm)
-   , (z(k),k=1,km+1), (ener(ig),ig=1,igm),emin,eneut
-   , (dampg(ig),ig=1,2*igm)
-
-     number of words = imsism+ism+jmsjsm+jsm+km+1+3*igm+2
-
-     x                  i-interval boundaries by i-set
-     y                  j-interval boundaries by j-set
-     z                  k-interval boundaries
-
-     ener               top energy boundary of group ig
-     emin               bottom energy boundary of group igm
-     eneut              bottom energy boundary of group neut
-                           (0 if neut=0)
-     dampg              restart acceleration damping data
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- cell zone identification:
-
-     ((ijzn(i,j),i=1,ims),j=1,jms)
-
-     number of words = sum of ims over j
-
-     ijzn               material zone by cell
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- cell scalar data:
-
-     ((flij(i,j),i=1,ims),j=1,jms)
-
-     number of words = sum of ims over j
-
-     flij               cell-average scalar flux
-
------------------------------------------------------------------------

      end
```

Table B.2. Format Descriptions For Direct-Access Scratch Files

```
------------------------------------------------------------------------
-
- name:     torscrf  (subject to change without notice)
-
- date:     20 mar 91
-
- purpose:  internal scratch file, cell flux or source moments
-
- notes:
-
-     this is a scratch file format, not a standard interface file.
-     the usual identification records are not present.  the user
-     must accept responsibility for identifying the source of the
-     file.
-
-     tort uses this format to store flux moments on ft91.  it
-     also uses this to store source moments on ft92 and ft93, except
-     that those files contain information for only the energy
-     group being calculated.
-
-     ft91 can be saved and used in a restart.  if ft94 is created,
-     it is also required for a complete restart.
-
-     njblk is adjusted by the code to best fit the problem into
-     memory, and its value is given in the output.  if njblk.eq.0,
-     ft91, ft92, and ft93 are not used.  if njblk.eq.1, then kl=1,
-     ku=km, and jb=1.  if njblk.gt.1, then the planes are grouped
-     into njblk blocks of jb planes each, where jb=(km-1)/njblk+1.
-     in this case, kl and ku become the first and last planes of
-     each block in turn, but no larger than km.
-
-     it may be noted that some systems impose physical record size
-     limitations, e.g. 131,072 words on unicos, and that may impose
-     a hidden level of blocking in subroutine dred.
-
- indices:
-           i is the first -dimension index.
-           j is the second-dimension index.
-           k is the third -dimension (plane) index.
-           l is the moment index.
-           ig is the group index, from high energy to low.
-
------------------------------------------------------------------------


------------------------------------------------------------------------
- file structure:
-
-      record type                          present if
-      -----------                          ----------
-
-  ............repeat for ig=1,igm
-  . . ........repeat for kb=1,njblk
-  . . ........repeat for k=kl,ku
-  . . . cell flux/source moments           njblk.ge.1
-  . . ........end k loop
-  . . ........end k-block loop
-  ............end ig loop
-
------------------------------------------------------------------------
```

```
------------------------------------------------------------------------
- cell flux/source moments:
-
-      ((((flum(i,l,j,k),i=1,im),l=1,lm),j=1,jm),k=kl,ku)
-
-      number of words = im*lm*jm*(ku-kl+1)
-
-      flum               cell-average moment data
-
------------------------------------------------------------------------
       end
```

```
-------------------------------------------------------------------------
-
- name:      torscrs  (subject to change without notice)
-
- date:      02 mar 1994
-
- purpose:  internal scratch file, boundary sources and fluxes
-
- notes:
-
-     this is a scratch file format, not a standard interface file.
-     the usual identification records are not present.  the user
-     must accept responsibility for identifying the source of the
-     file.
-
-     tort uses this format to store boundary source and flux
-     information on ft94.  it can be used with ft91 to give a
-     complete restart.
-
-     ifbnds.gt.0 if a fixed external boundary source is used.
-
-     ifbndf.gt.0 if boundary flux is to be saved for a periodic
-     boundary or an outer boundary with reflection.
-
-
- indices:
-           i is the first -dimension index
-           j is the second-dimension index
-           k is the third -dimension index
-           m is the discrete direction index
-           ig is the energy group index, from high energy to low.
-           ima = iabs(im), im is input
-           jm, km, igm are input
-           ims is the length of a given row, <=ima.
-           jms is the number of rows in a given plane, <=jm.
-           mms is the number of directions for group ig,
-             determined from input arrays.
-           mmsdu is the larger of the number of downward or
-             the number of upward directions for group ig.
-           the sko and fko arrays have either downward or upward
-             directions, padded with 0's to the required record
-             length.
-
-------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------
- file structure:
-
-      record type                          present if
-      -----------                          ----------
-
-  ............repeat for ig=1,igm
-  . . ........repeat for k=km,1,-1
-  . .   i,j-external boundary source   ifbnds.gt.0
-  . . ........end k loop
   .
-  . .   k-external boundary source, dn ifbnds.gt.0
-  . .   k-external boundary source, up ifbnds.gt.0
-  ............end ig loop
-
-  ............repeat for ig=1,igm
-  . . ........repeat for k=km,1,-1
-  . .   i,j-external boundary flux     ifbndf.gt.0
-  . . ........end k loop
   .
-  . .   k-external boundary flux, dn   ifbndf.gt.0
-  . .   k-external boundary flux, up   ifbndf.gt.0
-  ............end ig loop
-
-----------------------------------------------------------------------


-----------------------------------------------------------------------
- i,j-external boundary source:
-
-     ((sio(m,j),m=1,mms),j=1,jms),
-     ((sjo(m,i),m=1,mms),i=1,imx)
-
-     number of words = mms*(jms+imx)
-
-     imx    max of ims for inside and outside rows for this k
-     sio    i-boundary directional source
-     sjo    j-boundary directional source
-
-     this record has length 1 if nbc1.ne.4 .and. nbcr.ne.4 .and.
-     nbci.ne.4 .and. nbco.ne.4 .and. nifbnd.eq.0.
-
-----------------------------------------------------------------------


-----------------------------------------------------------------------
- k-external boundary source:
-
-     (((sko(m+m1,i,j),m=1,mmsdu),i=1,ims),j=1,jms)
-
-     number of words = mmsdu*sum ims over all j for this k
-
-     sko    k-boundary directional source
-     m1     0 for downward sources, else number of downward
-            directions
-
-     sko downward: length 1 if nbct.ne.4 .and. file is written by tort.
-     sko upward:   length 1 if nbcb.ne.4 .and. file is written by tort.
-
-----------------------------------------------------------------------


-----------------------------------------------------------------------
- i,k-external boundary flux:
-
-     ((fio(m,j),m=1,mms),j=1,jms),
```

```
-      ((fjo(m,i),m=1,mms),i=1,imx)
-
-      number of words = mms*(jms+imx)
-
-      imx     max of ims for inside and outside rows for this k
-      fio     i-boundary directional flux
-      fjo     j-boundary directional flux
-
-      this record has length 1 if
-        nbcl.ne.2 .and. nbcr.ne.1 .and. nbcr.ne.2 .and.
-        nbci.ne.2 .and. nbco.ne.1 .and. nbco.ne.2
-
-------------------------------------------------------------------------


-------------------------------------------------------------------------
- k-external boundary flux:
-
-      (((fko(m+m1,i,j),m=1,mmsdu),i=1,ims),j=1,jms)
-
-      number of words = mmsdu*sum ims over all j for this k
-
-      fko     k-boundary directional flux
-      m1      0 for downward fluxes,  else number of downward
-              directions
-
-      fko downward has length 1 if nbct.ne.2
-      fko upward   has length 1 if nbcb.ne.2 .and. nbct.ne.1
-
-------------------------------------------------------------------------
      end
```

# APPENDIX C. DETAILED INSTALLATION INSTRUCTIONS



INSTALLATION INSTRUCTIONS

for
CMP (Code Maintenance Package)
RSCORS GRAPHICS SYSTEM
ISOPLOT GRAPHICS POST-PROCESSOR
DORT, TORT, AND OTHER DISCRETE ORDINATE SYSTEM CODES

on
SUN, HP, SGI, IBM-RISC, DEC-ULTRIX, AND UNICOS SYSTEMS

prepared by

David B. Simpson
Computational Physics and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6363

CONTENTS

ACKNOWLEDGEMENT

This document is an edited version of an original document prepared by the late Sam Thompson of Sandia National Laboratory. With Sandia permission to redistribute the CMP, RSCORS, and related utilities, we wish to acknowledge that the aforementioned software was authored by

Samuel L. Thompson
Thermal/Hydraulic Analysis Department, 6418
Sandia National Laboratories
Albuquerque, New Mexico USA 87185

## INTRODUCTION

This set of installation instructions are for the CMP code maintenance package, the RSCORS graphics library, the ISOPLOT postprocessor for DORT, and the DORT, TORT, and other Discrete Ordinate System (DOS) codes on SUN, HP, SGI, IBM-RISC, DEC-ULTRIX, AND UNICOS computer systems.

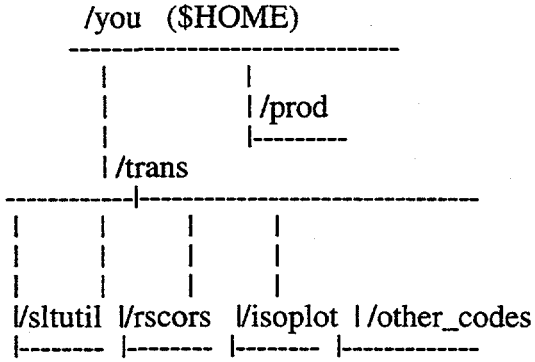
The utility programs support various SNL physics and engineering programs such as CHARTD, CSQ, CTH and MELCOR. These utilities must be installed before the physics codes and their graphics support programs, since some of them are used in the installation of the physics and engineering codes. Detailed descriptions of each of the programs are given in the code reference manual. All utility programs are written in Fortran 77 with a few C language routines.

There are two forms of the installation package included with the distribution files. The first form, described in this document, uses a single command for complete installation of all packages. The second form has separate utilities for each package and allows the user to modify the code. Instructions for the second form are found in the document INSTUTIL.DOC. In most cases this first form is adequate for the user with main interest in the physics and engineering codes.

## DIRECTORY STRUCTURE

The directory structure used during installation is shown below. The root directory for the working installation files and source code is named "trans". The directory "prod" is used to store executables and libraries for production applications. You can name either anything that you like. UNIX style directory names are used.

With some physics code distributions, these packages are supplied with different directory names but the same directory structure. The following still applies with modified names.



The CMP related programs are in /sltutil, the RSCORS related codes are in /rscors, and the isoplot codes are in /isoplot. The installation procedures build the executables and libraries in these directories. After compiling and linking is complete, the executables and libraries are copied to the /prod directory. The distribution files and directories can then be deleted to save file space if desired. A short description of each program is given later in this document. On some UNIX systems, you must run ranlib if you copy a library file (files with .a).

The executables and libraries created are

| directory | utility | | |
|---|---|---|---|
| /sltutil | CMP | cmpcopy | |
| | cmpdif | beep | |
| | mupack | t2p | |
| /rscors | librscors.a | poppst | |
| | popx11 | popx12 | |
| | poptk4 | popt15 | |
| | popalp | popt05 | |
| | polp | popsun | |
| /isoplot | isoplot | | |
| /dort | dort | bndrys | visa |
| | alc | grtuncl | torsed |
| | gip | rtflum | torset |
| /tort | tort | | |
| /Doc | documentation | | |
| /Install | shell scripts for installing | | |
| /bin | directory where executables are stored<br>Note - extra copy is in "prod" directory | | |

On UNIX system the "prod" directory should be in the path statement.

Note that "CMP" is in upper case while the rest are lower case. UNIX already has a cmp (a difference utility).

## FILE SPACE

There are a considerable number of files generated by the procedures. If you have file space problems, you can delete most of them. In general, the procedures do not delete files that you will need if there are problems. If everything works, you only need the executables and the RSCORS library.

## DISTRIBUTION TAPE FORMAT

SUN, HP, ULTRIX, and IBM-RISC: The tape was written from the /trans directory with the command

```
tar cvf /dev/rmt0 *
```

To read the tape, get in your "/trans" directory and type

```
tar xvf /dev/rmt0 (use name of your tape drive)
```

Tar will create the required subdirectories.

## INSTALLATION

1. Determine the names that you wish to use for the two installation directories. These have default names of /trans and /prod.

2. Create the directories and put the files in the directories under /trans. If the files are being read from tape, the subdirectories will be created by tar (UNIX).

   ```
   mkdir trans
   mkdir prod  (unless an existing directory is used)
   cd trans
   tar xvf ...
   ```

   If you are ftp-ing the files from another system, create the two subdirectories manually. Tar could also be used for the transfer. Depending on your requirements, you might set default file permissions.

3. If you want the Xwindow graphics drivers, you should locate the file libX11.a on your system. This is the MIT Xwindow driver and is used by the procedures at link time. The location of this file has been a problem mainly on sun systems. If this file is in a nonstandard location (for the compiler/linker default path), you can copy the file to /trans/rscors and execute ranlib.

If the procedures can not find the X11 library, the codes popx11 and popx12 will not link. See the log file if there is a problem. You can also skip the Xwindow installation during the execution of sinstall and return to this at a later time. See the file /trans/Install/mupopx11.sys.

4. Go to the /trans/Install directory and execute
   set path ( $path ../bin . ) if using csh,
   or PATH=$PATH:../bin:.;export PATH if using bsh or ksh
   sinstall.unx

   The procedure will ask several question and then complete the installation of all packages as a batch job. A log file is generated. If there is an abort, first check the file space then the log file.

   The system names used by the procedure are
   sun = SUN workstation
   ucs = CRAY UNICOS
   aix = IBM RISC 6000 workstation
   utx = DEC ULTRIX workstation
   hpw = Hewlett Packard UNIX workstation
   sgi = Silicon Graphics IRIX

   This completes the installation except for testing and symbol definitions discussed below.

---

TESTING

You should now test the execution of each package. If any of these tests fail, you will have to use the second installation method.

To test the CMP and ISOPLOT codes, go to the appropriate directory and type

CMP
cmpcopy
cmpdif
isoplot

and enter "QUIT" and return for each question. If the program begins execution without a strange system abort message, the files are usable.

Testing of the RSCORS libraries depends on which display options you are using. This will be covered at a later point.

## DOCUMENT FILES

Available documentation for each code is included in the Doc directory. The CMP manual is file CMP.DOC in /Doc. The RSCORS manual is file RSCORS.DOC in /Doc. Other graphics related documents are in this directory. These manuals are formatted with 66 lines per page with only ASCII text. They can be used as online help or printed using t2p.

The figures for the RSCORS manual are generated by FORTRAN programs supplied with this package. Generation of the figures will be discussed later.

## HOW RSCORS WORKS

The distributed version of RSCORS uses a metafile system. The graphics program (a fortran program which calls RSCORS) is linked to the RSCORS library. When the graphics program executes, the plot data is written to a local file called a metafile. The metafile can contain many graphics images.

This metafile is later read by one (or all) of the "POP" (Post Option Processor) utilities to actually display the data. The POP program will allow you to skip around in the file, edit the data, and do lots of wonderful things. The name POPxyz has been connected to an output device with name xyz. POPxyz has inline help for interactive questions.

The metafiles are computer system independent bit streams that can be transferred between different computers with binary file transfer. This includes PCs, workstations and main frames.

It is also possible for RSCORS programs to display data directly to the computer terminal as your code executes. A few additional routines are being constructed to make this easy under PC windows and workstation Xwindows. These are NOT included in this distribution at this time.

The following POP utilities are supplied:

a. POPX11 and POPX12 are Xwindow drivers for the MIT X11 release 4 systems. See note below.

b. POPPST translates a metafile to PostScript output format. You must send the output file POPOUT.PST to a PostScript plotter. Both Black & White and Color systems are supported.

c. POPTK4 displays metafile data in Tektronix 4014 format. This is a black & white format used by many PC modem packages.

d. POPT05 displays metafile data in Tektronix 4105 format. This is a 8 color format used by some PC modem packages.

e. POPT15 displays metafile data in Tektronix 4115/4125 format. This is a 256 color format.

f. POPALP generates a printer "plot" for checkout if none of the other display options will work. This is for location of problems only. The graphics is terrible but it will function on anything.

g. POPSUN is only available for older SUN systems running the SUNVIEW window system and is not generated by the default installation. (Xwindow systems should use POPX11 and POPX12.) POPSUN is linked to SUNVIEW using the SUNCORE graphics system supplied with the SUN OS by the mupopsun.sun procedure. If you have trouble with this, try the simple example of a FORTRAN code in the SUNCORE reference manual. SUNCORE will not run on all SUN systems - in particular, those with the expensive graphics frame buffer. I do not have a simple solution in this case - you could use the T15 driver with a Tektronix package on the SUN. ISOTEK seems to work well. Another solution, which you will get from the SUN software support group, might be the Xwindow package, i.e., buy the OpenWindows package, and use POPX11 and POPX12.

---

NOTES CONCERNING XWINDOW (MIT X11 RELEASE 4) SYSTEMS

This package is called "DecWindows" on the VMS and ULTRIX systems. On the SUN there is the SUN "OpenWindows" system. On UNICOS and IBM-RISC, it is called Xwindows. To use this package your systems must have the Xwindow package installed. If you have not used Xwindows before you should get help from your system manager. Although a best guess has been included in mupopx11.sys as where to look for the libraries and "C" include blocks, you might have to make some changes.

POPX11 and POPX12 both work in a local or remote mode. If you are using Xwindows to display graphics from a remote system (client/server), you must set the environment variable "DISPLAY" on the remote system and "xhost" on your workstation as with any Xwindow application. The exact form varies from system to system. If you see the message "svdi: cannot create window on ...", it means that the environment variables are not set correctly or your hardware will not support the function.

POPX11 and POPX12 differ in their treatment of color tables. POPX11 uses the system default color table. Colors might not be exactly as defined. In some cases, the system might not be able to get the desired colors and will give a message "svdi: problem getting color". In this case use POPX12. POPX12 redefines the system color table. Unfortunately, this will probably change the colors in other windows on your terminal if you use the full 256 colors allowed with Xwindows. Color "flashing" will be seen as you move the mouse around the screen. Colors will return to normal when the graphics window is closed.

POPX12 also allows optional double buffering of displayed data. The entire display is changed at the same time. This is required for animation. POPX11 does not include this feature.

---

## TESTING OF THE POP UTILITIES

During the installation a small metafile "test.met" was created in /rscors. You can test the execution of each of the POP utilities with this metafile. For a Xwindow system, type

```
popx12 test.met (or try popx11)
```

For each required input, if you do not understand the options, just enter return and a help display will appear. Detailed help related to file editing and other features is available.

To print output on a PostScript printer on a UNIX system, type

```
poppst test.met
lpr -Pprinter popout.pst
```

The term "BATCH" means process all pages in the metafile. The term "INTERACTIVE" means that you can select pages from a larger set.

---

## LINKING WITH RSCORS ON UNIX

On UNIX add the production directory "prod" to your path statement in .login, .profile, or .cshrc as appropriate. You must also define one environment variable.

In .cshrc

```
setenv RSCORS ..path../prod/librscors.a
```

In .profile

```
RSCORS=..path../prod/librscors.a
export RSCORS
```

You should logout and then login to get the symbols and paths defined.

To link your code to RSCORS, just add $RSCORS to your compiler command. For example, on a sun

```
f77 -o urcode urcode.f $RSCORS
```

or on IBM/RISC

```
xlf -o urcode urcode.f $RSCORS
```

Execution of urcode will create a metafile which is read by the "pop" utilities and displayed as appropriate.

---

## ILLUSTRATIONS FOR THE RSCORS REFERENCE MANUAL

The figures for the RSCORS reference manual are created by THREE fortran programs. These are FIGREPT1.FOR, FIGREPT2.FOR and FIGREPT3.FOR. They are normal fortran codes which are executed as any other program that uses the RSCORS library. The color and black/white display of these figures are quite different.

---

## NOTES ON OTHER UTILITIES

Four of the utilities supplied in the distribution package are not documented.

MUPACK concatenates files for shipment between computer systems and unpacks them on the second system. It works from a list of file names to pack. All packed files must be text files. Each directory contains a file named pack.lst used to move the entire directory to other computer systems. Mupack itself must be moved separately. Mupacked files can be moved over phone lines and between difference types of systems. Use tar for UNIX to UNIX transfers.

BEEP is a simple utility that speaks its name.

POLP is a 1024x1024 rasterizer for graphics metafiles. You will not need this until you generate plots that take an excessive amount of time to display (color shaded plots from csqplt or cthplt). POLP uses metafiles for both input and output. Use the workstation option to be started. POLP has inline help for interactive questions.

T2P is an ascii text to PostScript translator. It is useful for printing the code manuals (with 66 lines per page) and 80/132 column fortran code output in a compressed format. Type t2p without parameters for a complete list of options. T2P is only available on UNIX and PC-DOS systems.

INSTALLATION CHECK LIST

This is a summary of the installation steps.

1. Determine "sys" symbol for your system.

2. Go to, or create and go to, /trans and /prod.

3. Read distribution tape with UNIX tar or ftp files from another machine.

4. If Xwindows system, check X11 library locations.

5. Go to /trans/Install and type sinstall.unx (UNIX). Answer questions as approriate.

6. Wait until batch job completes. Read rest of instructions while waiting. Look for aborts in log file after completion of background job.

7. Go to the individual directories and test code.

8. Modify .login, .cshrc, .profile, or login.com as appropriate for your system.

9. Logout and login.

10. If you are going to use a remote Xwindow system (client/server), you must go through the same procedure on the other system and set the environment variables.

# APPENDIX D. DISCONTINUOUS SPACE MESH

## BASIC CONCEPT

In a conventional continuous-mesh problem, a computational mesh is defined by mesh-interval boundaries along each of the coordinate axes. Planes passing through these interval boundaries, perpendicular to the respective coordinate axes, define the surfaces of each cell. Opposite cell surfaces are always parallel, and they always meet adjacent surfaces at right angles. The cell surfaces run continuously through the mesh.

In such a mesh, let us call the first, second, and third coordinate axes the i, j, and k axes. The mesh cells lie in ordered rows parallel to the i axis, and the rows lie in planes perpendicular to the k axis. The vertical boundaries of each cell match the boundaries of adjacent cells.

In a discontinuous mesh, as the term is used here, the requirements are relaxed slightly. Mesh cells are still bounded by parallel planes, each perpendicular to one of the coordinate axes, and the planes meet at right angles. The new flexibility is that only the k boundaries, i.e. the boundary planes perpendicular to the k axis, are required to run continuously through the mesh. Thus, the other boundary planes may be discontinuous at intersections. The mesh cells lie in rows having common j boundaries, but their i boundaries need not match. Rows lie in planes sharing common k boundaries, but neither i nor j boundaries of adjacent planes need match (except as required to allow acceleration and at the outer boundaries of the problem space).

The set of boundaries perpendicular to the i axis for a given row is called an "i-set". Similarly, the set of boundaries perpendicular to the j axis for a given plane is called a "j-set". A problem has only one "k-set", of course, since the plane boundaries run continuously through the mesh.

The advantage is that the mesh can be locally dense in areas where detail is needed most, thus using computational work more efficiently. An illustration is provided in Fig. D.1 (pg. D-6). Since the transport within each cell is unperturbed by the irregularities, conventional evaluation procedures such as weighted difference, nodal, or characteristic methods can be used. Many years of research have gone into these methods, and they would not be relinquished easily.

The partial current acceleration already in widespread use in DORT and TORT is applicable to this type of mesh. The present implementation carries the restriction that a coarse mesh be supplied, and that the coarse-mesh boundaries lie in each of the fine-mesh sets. This restriction speeds evaluation of the acceleration matrix, and it also inhibits a type of mesh dispersion that might otherwise arise.

The programming is significantly more complicated with the discontinuous mesh, but a system of pre-calculated "pivot arrays" allows data items to be located and used without measurable loss of efficiency. The pivot arrays will be discussed later. Since all of the mesh cells lie in rows, the computational sweeps performed by the conventional TORT/DORT subroutines can be used without modification, and they will run at the traditional speeds. Some computational work is required, of course, to perform the "remeshing", i.e. the remapping of boundary flows where adjacent rows and planes do not match. TORT can produce the standard results to a conventional problem without cost penalty due to the discontinuous feature, however.

Probably the most important disadvantage of the discontinuous-mesh concept in two dimensions is that, although it can help in describing curved surfaces, it is not as powerful as general triangles or general quadrilaterals in this respect. Codes using the latter two concepts in discrete ordinates calculations exist, although none appear to have reached widespread use. It is not presently clear when these concepts will reach routine production use in three dimensions, or what the accuracy and efficiency would be.

## MACHINE IMPLEMENTATION

We must necessarily ask the reader to pardon a mix of FORTRAN and algebra in that which follows. We will try to be clear. First, we define:

IS = index of an i-set, i.e. a set of X or R boundaries defining mesh-cell surfaces; IS=1,...,ISM; and

JS = index of a j-set, i.e. a set of Y or $\Theta$ boundaries defining mesh-cell surfaces; IS=1,...,ISM.

For j-sets, input arrays consist of:

JMBJS(JS) = # of mesh cells in j-set JS , and

JSET(K) = j-set number for rows in plane K; K=1,...,KM .

From these, we can always obtain:

JS = JSET(K), index of the j-set for plane K , and

JMS = JMBJS(JS), number of intervals for j-set JS .

With regular indexing, where all JMBJS(JS) = JM, we (or the compiler) can locate a function of J and JS by a simple integer computation:

F(J,JS) = F(J+JM*(JS-1)) .

Since JMBJS is not necessarily constant in a discontinuous mesh, we now define a "pivot array":

JBJS(1) = 0 , and

JBJS(JS+1) = JBJS(JS) + JMBJS(JS); JS=1,...,JSM .

We denote the "irregular indexing" by (J'JS), rather than (J,JS). The item corresponding to J and JS can be found by:

F(J'JS) = F(J+JBJS(JS)) .

In general, this is as computationally efficient as the conventional method of indexing. It requires additional storage, but generally not enough to present difficulty.

In the case of arrays such as Y or $\Theta$, JMS rows are bounded by JMS+1 interval boundaries, and the use of the pivot array is slightly different. For example, the larger of the two Y's bounding interval J in j-set JS is located by:

$$Y(J'JS) = Y(J+JBJS(JS)+JS) \quad .$$

Now, we define a new pivot array that will be a bit more indirect in definition, but much more useful:

$$JBK(1) = 0 \quad , \text{and}$$

$$JBK(K+1) = JBK(K) + JMBJS(JSET(K)); \quad K=1,...,KM \quad .$$

From this a function of J and K can be obtained immediately:

$$G(J'K) = G(J+JBK(K)) \quad .$$

It is convenient to note that JMS can be obtained in several ways, depending upon which data happen to be at hand:

$$JMS = JMBJS(JSET(K)) \quad , \text{or}$$

$$= JBJS(JSET(K)+1) - JBJS(JSET(K)) \quad , \text{or}$$

$$= JBK(K+1) - JBK(K) \quad .$$

The overall number of rows is given by:

$$JMKM = \sum_{K=1}^{KM} JMBJS(JSET(K)) \quad , \text{or}$$

$$= JBK(KM+1) \quad .$$

This indexing scheme is the same method used in DOT 4 and DORT up to this point. The treatment of the i-mesh sets follows the same plan, but it is a bit messier, since it is nested one layer deeper. We use input arrays:

$$IMBIS(IS) = \#\text{ of mesh cells in the IS}^{th}\text{ i-set} \quad , \text{and}$$

$$ISET(J'K) = \#\text{ of the mesh set in the J}^{th}\text{ row of the K}^{th}\text{ plane} \quad .$$

Once again, we can obtain:

IS = ISET(J'K), the i-set used in row J of plane K , and

IMS = IMBIS(IS), the length of i-set IS .

Now, we define:

IBIS(1) = 0 , and

IBIS(IS+1) = IBIS(IS) + IMBIS(IS); IS=1,...,ISM ,

so that a function of I and IS can be located:

H(I'IS) = H(I+IBIS(IS)) .

For arrays such as R or X, IMS cells are bounded by IMS+1 interval boundaries. For example, the larger of the two R's bounding interval I in i- set IS is located by:

R(I'IS) = R(I+IBIS(IS)+IS) .

Now, we define a new pivot array in terms of a scalar variable, JK, that runs through each value of J for a plane, then through each plane in turn, plus a final terminating value; i.e.

JK=1,...,JMBJS(JSET(1)),JMBJS(JSET(1))+1,...,JMKM,JMKM+1 .

In terms of this variable, we now define:

IBJK(1) = 0 , and

IBJK(JK+1) = IBJK(JK) + IMBIS(ISET(JK)) .

From this, a function of I, J, and K that would be, with regular mesh:

P(I,J,K) = P(I+IM*((J-1)+JM*(K-1))) ,

becomes, with irregular indexing:

P(I'J'K) = P(I+IBJK(J+JBK(K))) .

We also define:

IJBK(1) = 0 , and

$$IJBK(K+1) = IJBK(K) + \sum_{J=1}^{JMBJS(JSET(K))} IMBIS(\,ISET(J+JBK(K)));\ K=1,...,KM \quad .$$

This is needed for indexing things that vary by I and J, but not K:

Q(I'J) = Q(I+IBJK(J+JBK(K))-IJBK(K)); K=constant .

We also note that IMS can be obtained variously by:

IMS = IMBIS(ISET(K)) , or

= IBIS(ISET(J+JBK(K))+1) - IBIS(ISET(J+JBK(K))) , or

= IBJK(J+JBK(K)+1) - IJBK(J+JBK(K)) ,

and the overall number of mesh cells is:

$$\text{IMJMKM} = \sum_{K=1}^{KM} \sum_{J-1}^{JMBJS(JSET(K))} \text{IMBIS(ISET(J+JBK(K)))} \quad \text{, or}$$

= IBJK(JMKM+1) , or

= IJBK(K+1) .

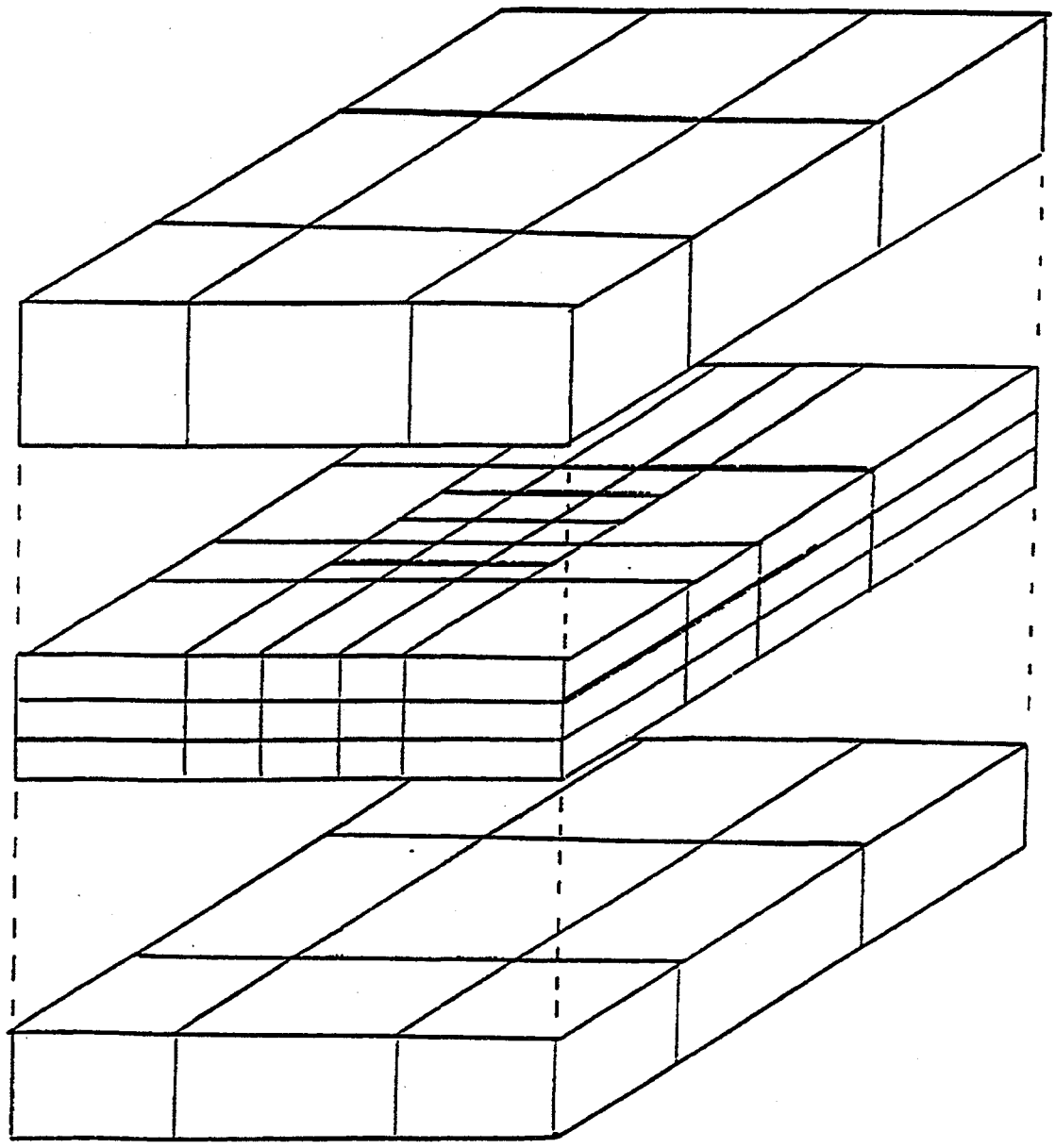


Fig. D.1. The Discontinuous Mesh Feature, Showing Locally Dense Mesh Detailing Problem Features

## APPENDIX E. ACCURACY OF LOW-ORDER APPROXIMATIONS

In certain simple situations, analytical solutions to Eq. (3.13) can be obtained readity, and it may be instructive to compare those with the results of the linear and step formulations. In 1-D slab geometry, Eq. (3.13) reduces to Eq. (3.50), which has the analytical solution Eq. (3.54). When the source is 0, the result has the form:

$$N(s) = N_0\, e^{-s} \; ; \; N_0 = N_{i-c} \quad , \tag{E1}$$

which, with the exponential expanded, is equivalent to:

$$N(s) = N_0\left(1 - s + (1/2)s^2 - (1/6)s^3 + (1/24)s^4 + ----\right) \quad , \tag{E2a}$$

where:

$$s \equiv \sigma^T (x - x_{i-c})/\mu \quad . \tag{E2b}$$

Since $N(s)$ is the flux emerging from an interval of width $s$, we can compare this result with the estimates of $N_{i+1}$ made by our difference models. The linear difference model for this restricted case can be found by setting terms in $B$, $C$, $\Delta A$, and $S$ in Eq. (3.23) to 0. With this and Eq. (3.18a), for $\mu>0$:

$$N = \frac{2\mu A N_0}{V\sigma^T + 2\mu A} = \frac{N_0}{(1/2)V\sigma^T/\mu A + 1} \tag{E3a}$$

$$N_{i+c} = 2N - N_0 \quad . \tag{E3b}$$

Observing that, at $x=x_{i+1/2}$, $V=A\Delta x$:

$$N = \frac{N_0}{1 + (1/2)s} \tag{E4a}$$

and

$$N_{i+c} = N_0 \left( \frac{1 - (1/2)s}{1 + (1/2)s} \right) \tag{E4b}$$

Expanding this in a Maclaurin series:

$$f(x) = f(0) + f'(0)x + (1/2)f''(0)x^2 + (1/6)f'''(0)x^3 + --- \quad , \tag{E5}$$

we have:

$$N_{i+c} = N_0\left(1 - s + (1/2)s^2 - (1/4)s^3 + \text{---}\right) \quad . \tag{E6}$$

Comparing this with Eq. (E2), we see that it is correct through the term in $s^2$.

For the step model, we use Eq. (3.27a) and Eq. (3.26a), giving:

$$N_{i+c} = N = \frac{N_0}{1+s} \quad . \tag{E7}$$

The Maclaurin series expansion of this is:

$$N_{i+c} = N_0\ (1 - s + s^2 + \text{---}) \quad , \tag{E8}$$

which is correct only through the term in $s$. For small $s$, then, the step method is inherently inferior to the linear method, as was found in an earlier section. For large $s$, however, the linear-method result approaches $-N_0$, while the step result approaches 0, as it should.

Another case of importance is that with an internal source, but no boundary flow. Assuming a uniform source, $Q$, and no boundary flow in Eq. (3.54):

$$N(s) = \frac{Q}{\sigma^T}\left(1 - e^{-s}\right) \quad , \tag{E9}$$

for which the series expansion is:

$$N(s) = \frac{Q}{\sigma^T}\ \left(s - (1/2)s^2 + (1/6)s^3 + \text{---}\right) \quad . \tag{E10}$$

For these assumptions, the linear model used in Eq. (3.23) becomes:

$$N = \frac{VQ}{V\sigma^T + 2\mu A} = \frac{(1/2)\,VQ/\mu A}{1 + (1/2)\,V\sigma^T/\mu A} \quad . \tag{E11}$$

In terms of $s$,

$$N = \frac{Q}{\sigma^T}\left(\frac{(1/2)s}{1 + (1/2)s}\right) \quad , \tag{E12a}$$

and, using Eq. (E3b):

$$N_{i+c} = \frac{Q}{\sigma^T} \left( \frac{s}{1 + (1/2)s} \right) \quad . \tag{E12b}$$

This expansion can be written by comparison with Eqs. (E7) and (E8):

$$N_{i+c} = \frac{Q}{\sigma^T} \left( s - (1/2)s^2 + (1/4)s^3 + \text{---} \right) \quad . \tag{E13}$$

The linear model is correct through the term in $s^2$ in this source-driven case, as it was in the boundary-driven case.

Examining the step result, Eq. (3.27a) becomes:

$$N_{i+c} = N = \frac{VQ}{V\sigma^T + \mu A} = \frac{VQ/\mu A}{1 + V\sigma^T/\mu A} \quad . \tag{E14}$$

This is equivalent to:

$$N_{i+c} = \frac{Q}{\sigma^T} \left( \frac{s}{1 + s} \right) \quad , \tag{E15}$$

which has the expansion:

$$N_{i+c} = \frac{Q}{\sigma^T} \left( s - s^2 + s^3 + \text{---} \right) \quad , \tag{E16}$$

and is correct only through the term in $s$. For large $s$, however, the step model approaches the correct value, while the linear model goes to the twice the correct value.

Now, let us look at a linearly varying source in the form:

$$Q = Q_1 s \quad . \tag{E17}$$

Referring, again, to Eq. (3.54), we have the analytical solution:

$$N(s) = \frac{e^{-s}}{\sigma^T} \int_0^s e^{s'} Q_1 s' ds' \quad . \tag{E18}$$

From this:

$$N(s) = \frac{Q_I}{\sigma^T}\left[s - \left(1 - e^{-s}\right)\right] \quad , \tag{E19a}$$

which has the expansion:

$$N(s) = \frac{Q_I}{\sigma^T}\,(1/2)\left(s^2 - (1/3)s^3 + (1/12)s^4 + \text{---}\right) \quad . \tag{E19b}$$

Since the low-order approximations do not provide for a varying source over an interval, we must use the average source in the formulations:

$$Q_A = (1/2)\,Q_I s \quad , \tag{E20}$$

which, with Eq. (E13), gives a linear-model result for this source of:

$$N_{i+c} = \frac{Q_I}{\sigma^T}\,(1/2)\left(s^2 - (1/2)s^3 + \text{---}\right) \quad . \tag{E21}$$

This is, once again, correct through the term in $s^2$, at least insofar as the use of the average source is valid.

The step model, Eq. (E16), gives, for this source:

$$N_{i+c} = \frac{Q_I}{\sigma^T}\,(1/2)\left(s^2 - s^3 + \text{---}\right) \quad , \tag{E22}$$

which is also correct through the term in $s^2$.

The use of the average source is of questionable value in this comparison, but it is not valid at all to determine behavior for large $s$. The simple difference models used here can not treat such a case. The nodal and characteristic models, discussed elsewhere, can treat this case accurately, however.

In each of the cases studied above, the linear model provides the best accuracy as $s$ approaches 0. In each case for which results for large $s$ have validity, however, that model produces large errors for large $s$, while the step model approaches the correct result. Since practical problems frequently have $s$ unity or larger, especially along oblique paths, behavior in the large is of considerable practical concern. If scalar flux is negative and not trivially small, the result is obviously wrong, and it is likely to be harmful. If the scalar flux is not negative, it may, nonetheless, be badly in error due to the inclusion of negative values. Non-physical behavior in the large can also thwart stability by producing arbitrary results that change with each iteration.

The cases cited above are more extreme than the usual thick-interval problem. More often, neither the boundary flux nor the source is zero. Accordingly, let us examine the case where boundary flux and internal source are more nearly in balance:

$$\frac{Q}{\sigma^T} = \theta N_0 \quad , \tag{E23}$$

where $\theta \sim 1$. Equations (E1) and (E9) indicate a slowly-varying solution for such a case, even for large *s:*

$$N(s) = N_0 \left[ e^{-s} + \theta \left( 1 - e^{-s} \right) \right] \quad . \tag{E24}$$

Such problems are common when mutual scattering between directions or groups is large compared with the external source and destruction terms. Such situations occur in charged-particle transport and in thermal neutron propagation. It is simply not practical to limit the mesh size to a fraction of a mean free path in such cases. A practical rule based upon flux change from interval to interval is necessary.

The data in Table E1 give the emerging flux due to a uniform unit source, from Eq. (E9), together with the linear- and step-approximation results from Eqs. (E12b) and (E15). Both approximations are positive this time, but the linear model goes to twice the proper value for large *s*.

Since Table 3.2 gives the comparable data for unit boundary source, these can be combined to give the total solution for various values of $\theta$. One sees immediately that the solution for $\theta$=1 is exact for either model. Table E2 gives the result for $\theta$=0.8 and 1.2. Here, it is seen that the flux varies across the interval by no more than 20%, but the linear model gives errors as large as 20%. Worse yet,the zero fixup would be of no help to the linear model. Solving the flux and source parts separately with the zero fixup, and then combining them, would also fail. The step model gives an error no larger, at worst, than 4%, however.

**Table E1. Emerging Flux vs Interval Width, *s*, for Uniform Source**

| *s* | Linear | Exact | Step |
|---|---|---|---|
| 1 | 0.67 | 0.63 | 0.50 |
| 2 | 1.00 | 0.86 | 0.67 |
| 4 | 1.33 | 0.98 | 0.80 |
| 8 | 1.60 | 1.00 | 0.89 |
| ∞ | 2.0 | 1.0 | 1.0 |

**Table E2. Emerging Flux vs. Interval Width, *s*, for Uniform Source Plus Boundary Flow**

| θ | *s* | Linear | Exact | Step |
|---|---|---|---|---|
| .8 | 1 | .87 | .87 | .90 |
| | 2 | .80 | .83 | .87 |
| | 4 | .73 | .80 | .84 |
| | 8 | .68 | .80 | .82 |
| | ∞ | .60 | .80 | .80 |
| 1.2 | 1 | 1.13 | 1.13 | 1.10 |
| | 2 | 1.20 | 1.17 | 1.13 |
| | 4 | 1.26 | 1.20 | 1.16 |
| | 8 | 1.32 | 1.20 | 1.18 |
| | ∞ | 1.40 | 1.20 | 1.20 |

Note: θ is the ratio of source to the source required for a flat flux solution.

## APPENDIX F. THE $C_n$ FUNCTIONS AND THEIR APPLICATIONS

In the solution of first-order differential equations, one starts with an equation of the form:

$$\frac{d}{dt} N = f\left(N, x_1, x_2, - - -\right) \quad , \tag{F1}$$

where $f$ is, in general, a complicated function of $N$ and various "state variables," $x_1$, $x_2$, etc. If $N$ is a major contributor to $f$, the equation can be very difficult to solve in this form by direct numerical evaluation. Small numerical errors in the estimate of $N$ can mask the effects of the state variables. This can require the use of very small increments in t.

It often happens, however, that $f$ is almost linear in $N$, and so the equation can be "quasi-linearized":

$$\frac{d}{dt} N + \lambda N = g\left(x_1, x_2, - - -\right) \quad , \tag{F2}$$

where $g$ varies, at most, slowly with $N$. If $g$ can not be written explicitly, Cohen's extension of the Runge-Kutta procedure can be used for a step-by-step evaluation.[27] If $g$ can be written as an explicit function of $t$, however, a complete solution can be written at once. Approximating $g$ as:

$$g = g_0 + g_1 t + g_2 t^2 + - - - \quad , \tag{F3}$$

Equation (F2) can be solved by use of an integrating factor:

$$\frac{d}{dt}\left(N e^{\lambda t}\right) = e^{\lambda t}\left(g_0 + g_1 t + g_2 t^2 + - - -\right) \quad ,$$

and

$$N(t) = N_0 e^{-t} + g_0 I_0(t) + g_1 I_1(t) + - - - \quad , \tag{F4a}$$

where:

$$I_n(t) \equiv e^{-t} \int_0^t s^n e^s ds \quad . \tag{F4b}$$

Integrating by parts, we can develop a recursion rule:

$$I_n(t) = e^{-t}\left[s^n e^s\right]_0^t - n\, e^{-t} \int_0^t s^{n-1} e^s\, ds \quad ,$$

$$= t^n - n\, e^{-t} \int_0^t s^{n-1} e^s\, ds \quad , \tag{F5}$$

$$= t^n - n\, I_{n-1}(t) \quad .$$

Now, for convenience, we define the $C_n$, functions in terms of this integral:

$$C_n(t) = \frac{1}{t^{n+1}} I_n(t) \quad , \tag{F6}$$

and the recursion becomes:

$$C_n(t) = \frac{1}{t}\left(1 - n\, C_{n-1}(t)\right) \quad , \tag{F7}$$

Given starting values, this recursion completely defines the $C_n$'s. We observe from Eqs. (F4) and (F6) that:

$$t\, C_0(t) = I_0(t) = 1 - e^{-t} \quad ,$$

and

$$C_0(t) = \frac{1}{t}\left(1 - e^{-t}\right) \quad . \tag{F8}$$

From this, we find:

$$C_0(0) = 1 \tag{F9a}$$

and

$$C_0(\infty) = 0 \quad . \tag{F9b}$$

From Eqs. (F6) and (F7), we find the limits for $C_n$:

$$C_n(0) = 1/(n+1) \tag{F10a}$$

$$C_n(\infty) = 0 \tag{F10b}$$

Looking at the first few $n$'s, we have:

$$C_0(x) = \left(1 - e^{-x}\right)/x = -\frac{1}{x}\left(e^{-x} - 1\right) \ , \tag{F11a}$$

$$C_1(x) = \frac{1}{x}\left[1 + \frac{1}{x}\left(e^{-x} - 1\right)\right] = \frac{1}{x^2}\left(e^{-x} - 1 + x\right) \ , \tag{F11b}$$

$$C_2(x) = \frac{1}{x}\left[1 - \frac{2}{x^2}\left(e^{-x} - 1 + x\right)\right] = -\frac{2}{x^3}\left(e^{-x} - 1 + x - (1/2)x^2\right) \ , \tag{F11c}$$

$$C_3(x) = \frac{1}{x}\left[1 - \frac{6}{x^3}\left(e^{-x} - 1 + x - \frac{1}{2}x^2\right)\right] = \frac{6}{x^4}\left(e^{-x} - 1 + x - \frac{1}{2}x^2 + \frac{1}{6}x^3\right) \ , \tag{F11d}$$

but, from the expansion for $e^{-x}$, we observe that:

$$e^{-x} = 1 - x + (1/2)x^2 - (1/6)x^3 + (1/24)x^4 - \ - - - \tag{F12}$$

Thus, it is evident that the function $C_n(x)$ is simply the exponential function with its n+1 leading terms deleted and the remaining terms renormalized. For this reason, the $C_n$'s are sometimes spoken of as "incomplete exponentials."

In principle, expansions of the sort shown in Eqs. (Flla-d) can be used for numerical evaluation, and this is generally satisfactory for t not too close to 0. Strong cancellation of terms is evident near $t$=0, however. In transport work, this includes the cases of very small intervals and rarefied material such as air. In time-dependent work, the problem has been dealt with by using the library evaluation of the exponential term above a certain small value of $t$ and a series expansion below. The exponential term is expanded according to Eq. (F12), and then, terms of like power in $t$ are collected. Other treatments are also possible.[28]

The library exponential functions are generally too expensive for transport work, however, and the use of a test on $t$ is impractical. Fortunately, high accuracy is needed over only a limited range for this application. Walters[29] has used the Padé(2,3) approximation to the exponential with considerable success. It has the form:

$$\frac{a_0 + a_1 x + a_2 x^2}{b_0 + b_1 x \ b_2 x^2 + b_3 x^3} \ .$$

This form can be substituted for the exponential in the expansions of Eq. (F11a-d), or new expansions of this general type can be fit. In either case, the result goes smoothly to the correct limit at $t$=0, it is quite sufficiently accurate for $t$ on the order of 1, and it goes smoothly to 0 for very large $t$.

The $C_n$ functions have some special properties that can be helpful. First, let us look at the $t$ moments:

$$M_n^m \equiv \int_0^t s^{n+1+m}\, C_n(s)\, ds \tag{F13}$$

The first two moments are readily evaluated:

$$\begin{aligned} M_0^0(t) &= \int_0^t s\, C_0(s)\, ds = \int_0^t \left(1 - e^{-s}\right) ds \\ &= e^{-2} - 1 + t \\ &= t^2\, C_1(t) \quad , \end{aligned} \tag{F14}$$

and

$$\begin{aligned} M_0^1 &= \int_0^t s^2\, C_0(s)\, ds = \int_0^t s\left(1 - e^{-s}\right) ds \\ &= (1+t)e^{-t} - 1 + (1/2)t^2 \quad . \end{aligned} \tag{F15}$$

Other moments can be found through a "Reduction Rule":

$$\begin{aligned} M_n^m(t) &= \int_0^t s^{n+1+m}\, C_n(s)\, ds = \int_0^t s^{n+m}\, ds - n \int_0^t s^{n+m}\, C_{n-1}(s)\, ds \\ &= \frac{1}{n+1+m}\, t^{n+1+m} - n\, M_{n-1}^m(t) \quad . \end{aligned} \tag{F16}$$

Applying this:

$$M_1^0(t) = \int_0^t s^2 C_1(s)ds = (1/2)t^2 - M_0^0(t)$$

$$= -\left(e^{-t} - 1 + t - (1/2)t^2\right) \qquad \text{(F17)}$$

$$= \frac{x^3}{2} C_2(t) \quad ,$$

and

$$M_1^1 = \int_0^t s^3 C_1(s)ds = (1/3)t^3 - M_0^1(t)$$

$$= -\left[(1+t)e^{-t} - 1 + (1/2)t^2 - (1/3)t^3\right] \qquad \text{(F18)}$$

One can also write a "Derivative Rule" and an "Integral Rule" by starting with *I:*

$$\frac{d}{dt} I_n(t) = \frac{d}{dt}\left(e^{-t}\int_0^t s^n e^s ds\right) = -e^{-t}\int_0^t s^n e^s ds - t^n$$

$$= -\left[t^n + I_n(t)\right] \quad , \qquad \text{(F19)}$$

and, from this:

$$\int_0^t I_n(s)ds = \int_0^t t^n ds - \left[I_n(s)\right]_0^t$$

$$= \frac{1}{n+1}\left[t^{n+1} - (n+1)\, I_n(t)\right] \quad . \qquad \text{(F20)}$$

Using Eq. (F5):

$$\int_0^t I_n(s)ds = \frac{1}{n+1} I_{n+1}(t) \quad . \qquad \text{(F21)}$$

$$\frac{d}{dt}\, t^{n+1}\, C_n(t) \;=\; -\, t^{n}\Big[1 + t\, C_n(t)\Big] \tag{F22}$$

and

$$\int_0^t s^{n+1}\, C_n(s)\, ds \;=\; \frac{1}{n+1}\, t^{n+2}\, C_{n+1}(t) \quad . \tag{F23}$$

One can see that Eqs. (F14) and (F17) are special cases of the Integral Rule.

# APPENDIX G. DISPERSION

The treatment of problems dominated by large regions of low-density material appears to have received less attention than that of penetration through thick scattering regions, and yet this is a very important class of problems for discrete ordinates codes. It is well known that certain incidental internal voids, especially void cavities around reactor vessels, can be treated successfully, given enough directions in the quadrature, but new problems arise when the region is as extensive as a room or building interior. In such problems, mesh sizes must typically be large, and much of the interior is filled with air. Particles typically scatter from surface to surface without penetrating deeply. Ray effects and artificial dispersion are potential difficulties.

While ray effects are well known, if not well controlled, artificial dispersion is not. The discussion relating to Fig. 3.2 posed the physical problem: radiation streaming past an opaque region can be artificially diverted into the shade volume by the numerical formulation alone. This effect is independent of mesh size, and it takes place whether scattering material is present or not. It can cause particularly severe errors when dealing with gammas, since their low albedos lead to volumes of deep shade. Flux can be much too high, too low, or even negative in the shade, and this can lead to serious errors in the direct beam as well.

The simple geometry of Fig. G.1 suffices to demonstrate and test for artificial dispersion. A square mesh in XY geometry has a monodirectional boundary source in one interval along the X axis. The radiation should flow between well-defined diagonal boundaries as shown in the figure. The analytical solution can be written by inspection. The mesh size is arbitrary, because the interior of the mesh is void. In this case, a mesh spacing of 1 and an incident flux of 100 are used.

In addition to the incident angle of 45° shown in Fig. G.1, 63° and 27° incidences will be considered, as shown in Figs. G.2 and G.3. In all of these problems, the average zone flux should be 0, 25, 50, or 75. As each numerical formulation is compared, we tabulate the largest error in any zone for which the correct result is 0, then 25, etc. Tables G1-G3 show the results of that comparison.

It happens that the incident direction of 45 degrees is a special case, in that the linear (diamond) method can solve this problem exactly without fixups, and the weighted difference result is also exact. One notes that the flux is constant along each boundary, either 0 or 100, and these methods have no problem with it. The simple step model, however, disperses so much of its flux into the 0-flux regions that the in-beam flux is very deficient.

Several nodal results are shown: safe nodal (WNODAL=0) has enough constraint on boundary flux

shapes to produce positive results in 3D problems; aggressive nodal (WNODAL=1) guarantees positive boundary flux shapes in 2-D problems, but it could, theoretically, give local negatives in 3-D; unconstrained nodal uses a special code modification to apply the linear boundary expansion without restraint; and constant nodal forces a constant boundary flux. Unconstrained nodal solves the problem exactly. All of the other nodal methods show errors of 12% or larger in the 0-flux zone, and this means that a factor-of-ten shading would be impossible with these methods.

The standard characteristic method produces a perfect result in the 0-flux zones and a better result than any of the constrained nodal methods elsewhere. Forcing the boundary flux shape to be a constant produces the exact result in all zones.

Turning to the more nearly straight-ahead case of Table G2 and Fig. G2, we see large errors for the previously-accurate linear and weighted methods, with so much dispersion into the shade areas that the in-beam flux is deficient by about 30%, just as was found with the step method at 45 degrees. In this problem, the flux along four of the vertical boundaries is 0 over half of the interval and 100 over the other half. The diamond-related methods can not represent this situation well at all. The constant nodal and constant characteristic methods fail similarly.

The constrained nodal methods, safe and aggressive, are approximately as accurate here as they were for the 45° case. The linear characteristic produces factor-of-ten shading in the 0-flux zone, but it has a higher error than the best nodal method elsewhere. The unconstrained nodal method produces a significant negative flux.

The oblique incidence of Fig. G3 and Table G3 has 3 horizontal surfaces with half-0 and half-100 flux. Once again, the diamond-related methods can not negotiate this problem at all. The unconstrained nodal method produces a negative result again. The aggressive nodal and linear characteristic methods both give fine accuracy on this problem.

Table G4 summarizes the worst error for any of the three previous cases. Of the positive methods, linear characteristic offers the least overall error. Aggressive nodal is more accurate in the direct-beam zones, and less accurate elsewhere. Either would give much better overall performance than the step, linear, or weighted methods on a dispersion-type problem.

It may be noted that a special version of the characteristic method was constructed in which the flux along each boundary was represented as two constant segments. This method solved all of these problems perfectly. While this particular formulation may not prove to be a practical general method, it indicates the flexibility that the characteristic approach offers for future experimentation.

Since no scattering or removal takes place in this problem, the mesh size is not relevant in the usual sense; there is no mean free path with which to compare the mesh size. Yet, one can seek to improve the boundary representation within the framework of the existing coding by subdividing each mesh cell, thus representing each boundary flux by several segments. This was done in the data presented in Table G5 for several of the numerical methods. The "3 X 3" mesh is identical to the geometry of Table G3. The other meshes represent successive halving of the spacing in each direction. All results are improved by the mesh refinement. The linear characteristic method has the best accuracy for the coarser meshes, although the aggressive nodal method is better for very fine meshes.

Since these original tests, a new nodal method has been added to the code. It does not require a choice between safe and aggressive restraints, and it seems to combine some of the best characteristics of both.

Of the methods tested here, linear characteristic gives the best overall performance if a positive method is required. Aggressive nodal performs slightly better if some risk of negatives can be accepted. Curiously, unconstrained nodal has the same maximum error as aggressive nodal, and it would not appear to offer benefits to compensate for its larger risk of negatives.

**Table G1. Error (%) vs. Method and Correct Zone Flux, Incident Angle=45°**

| | Correct Zone Flux | | | |
|---|---|---|---|---|
| Method | 0 | 25 | 50 | 75 |
| Step | 25 | -- | -31 | -- |
| Linear | 0 | -- | 0 | -- |
| Weighted | 0 | -- | 0 | -- |
| Safe Nodal | 16 | -- | -16 | -- |
| Aggressive Nodal | 12 | -- | -13 | -- |
| Unconstrained Nodal | 0 | -- | 0 | -- |
| Constant Nodal | 16 | -- | -20 | -- |
| Linear Characteristic | 0 | -- | 8 | -- |
| Constant Characteristic | 0 | -- | 0 | -- |

**Table G2. Error (%) vs. Method and Correct Zone Flux, Incident Angle=63°**

| | Correct Zone Flux | | | |
|---|---|---|---|---|
| Method | 0 | 25 | 50 | 75 |
| Step | 30 | 19 | -- | -45 |
| Linear | -15 | 39 | -- | -31 |
| Weighted | 17 | 14 | -- | -30 |
| Safe Nodal | 13 | 7 | -- | -18 |
| Aggressive Nodal | 11 | 5 | -- | -9 |
| Unconstrained Nodal | -6 | 13 | -- | -35 |
| Constant Nodal | 23 | 16 | -- | -35 |
| Linear Characteristic | 10 | 7 | -- | -15 |
| Constant Characteristic | 19 | 12 | -- | -31 |

**Table G3. Error (%) vs. Method and Correct Zone Flux, Incident Angle=27°**

| | Correct Zone Flux | | | |
|---|---|---|---|---|
| Method | 0 | 25 | 50 | 75 |
| Step | 15 | -10 | -28 | -- |
| Linear | -15 | 19 | -6 | -- |
| Weighted | 17 | -14 | -17 | -- |
| Safe Nodal | 11 | -5 | -11 | -- |
| Aggressive Nodal | 6 | -4 | -6 | -- |
| Unconstrained Nodal | -6 | 3 | -2 | -- |
| Constant Nodal | 14 | -7 | -17 | -- |
| Linear Characteristic | 5 | -2 | -4 | -- |
| Constant Characteristic | 12 | -6 | -13 | -- |

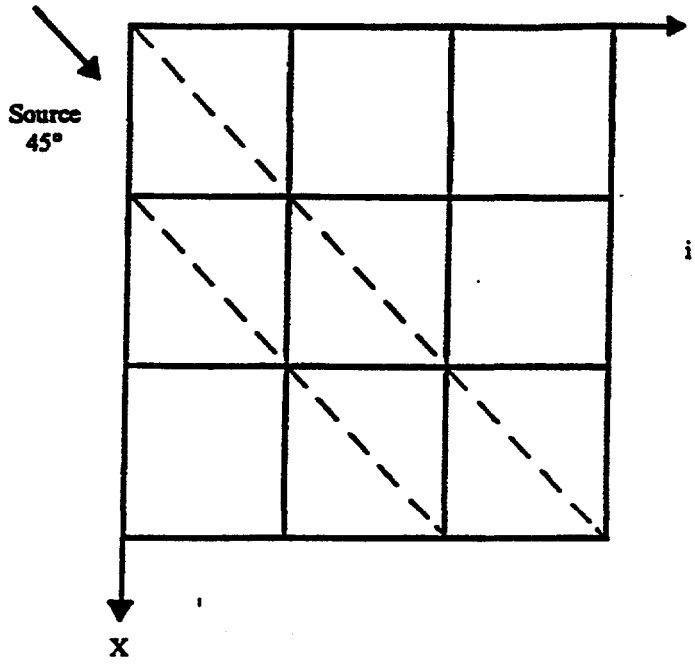
**Table G4. Worst Error (%) vs. Method and Correct Zone Flux, Incident Angles=45°, 63°, 27°**

| | Correct Zone Flux | | | | |
|---|---|---|---|---|---|
| Method | 0 | 25 | 50 | 75 | Overall |
| Step | 30 | 19 | -31 | -45 | -45 |
| Linear | -15 | 39 | -6 | -31 | 39 |
| Weighted | 17 | -14 | -17 | -30 | -30 |
| Safe Nodal | 16 | 7 | -16 | -18 | -18 |
| Aggressive Nodal | 12 | 5 | -13 | -9 | -13 |
| Unconstrained Nodal | -6 | 13 | -2 | -6 | 13 |
| Constant Nodal | 23 | 16 | -20 | -35 | -35 |
| Linear Characteristic | 10 | 7 | 8 | -15 | -15 |
| Constant Characteristic | 19 | 12 | -13 | -31 | -31 |

**Table G5. Error vs. Method, Correct Zone Flux, and Mesh Refinement, Incident Angle=27°**

| | | Correct Zone Flux | | | |
|---|---|---|---|---|---|
| Mesh | Method | 0 | 25 | 50 | Maximum, All Zones |
| 3X3 | WD | 17 | -14 | -17 | -17 |
| | SN | 11 | -5.0 | -11 | -11 |
| | AN | 5.6 | -3.6 | -5.6 | -5.6 |
| | LC | 5.4 | -2.0 | -4.2 | 5.4 |
| 6X6 | WD | 7.9 | -10 | -7.9 | -10 |
| | SN | 6.1 | -1.3 | -4.5 | 6.1 |
| | AN | 4.4 | 2.3 | -2.3 | 4.4 |
| | LC | 2.7 | 1.4 | -2.0 | 2.7 |
| 12X12 | WD | 3.1 | -6.4 | -3.0 | -6.4 |
| | SN | 1.7 | 0.8 | -1.4 | 1.7 |
| | AN | 1.1 | 0.7 | -0.8 | 1.1 |
| | LC | 1.0 | 1.4 | -0.6 | 1.4 |
| 24X24 | WD | 1.4 | 2.8 | -1.4 | 2.8 |
| | SN | 0.6 | 0.3 | -0.4 | 0.6 |
| | AN | 0.4 | 0.3 | -0.2 | 0.4 |
| | LC | 0.3 | 0.8 | -0.1 | 0.8 |

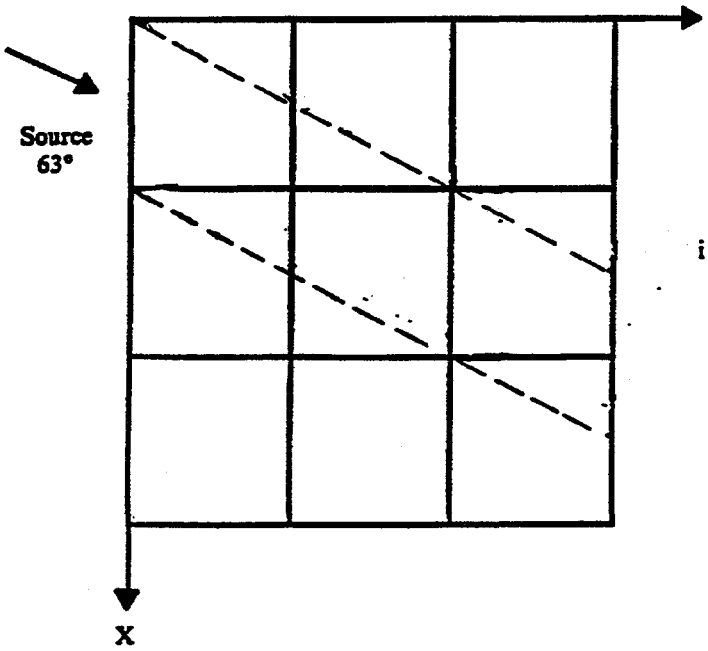
Notes:
1. The 3 X 3 mesh corresponds to Fig. G3. The 6 X 6 mesh is twice as fine, etc.
2. WD = weighted difference;
   SN = safe nodal;
   AN = aggressive nodal;
   LC = linear characteristic.




| | J = 1 | J = 2 | J = 3 |
|---|---|---|---|
| i = 1 | 50 | 0 | 0 |
| 2 | 50 | 50 | 0 |
| 3 | 0 | 50 | 50 |

$\mu$ = .57735
$\xi$ = .57735
$\eta$ = .57735

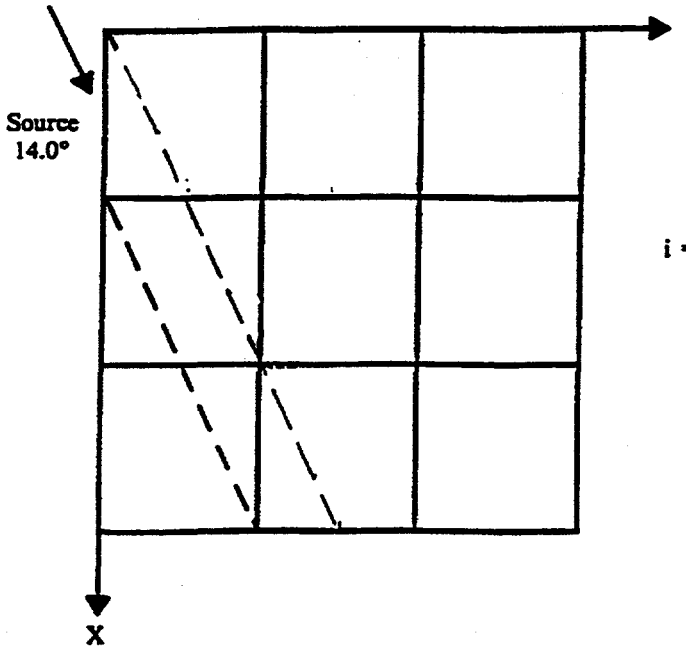
Fig. G.1. Geometry and Correct Result for Problem 1.




| | J = 1 | J = 2 | J = 3 |
|---|---|---|---|
| i = 1 | 75 | 25 | 0 |
| 2 | 25 | 75 | 75 |
| 3 | 0 | 0 | 25 |

$\mu$ = .36515
$\xi$ = .73030
$\eta$ = .57735

Fig. G.2. Geometry and Correct Result for Problem 2.




| | J = 1 | J = 2 | J = 3 |
|---|---|---|---|
| i = 1 | 25 | 0 | 0 |
| 2 | 50 | 0 | 0 |
| 3 | 25 | 25 | 0 |

$\mu$ = .73030
$\xi$ = .36515
$\eta$ = .57735

Fig. G.3. Geometry and Correct Result for Problem 3.

# APPENDIX H. SUBROUTINE LIST AND DESCRIPTION

Note: structure of routine listings is as follows:

1. routine name/entries (only important entries shown)
2. calling routines (unless called from many places)
3. description of function

## H.1 Main Structural Routines

| | | |
|---|---|---|
| tort | | container array sizing; calls to mesage, loco, dotx |
| mesage | | message printing at beginning of job |
| loco | | parameter input and edit |
| dotx | | execution coordination; final wrapup |
| input | (dotx) | pointer setup, array input, checking, editing |
| inpa | (input) | assistance to input in array setup |
| work | (dotx) | coordination of flux calculation; calls sorsum, fscon, source |
| sorsum | (work) | source summing and normalization |
| fscon | (work) | controls source iteration and acceleration |
| source | (work) | sweep of all groups for a given source |
| flux | (source) | flux iteration and acceleration control |
| group | (flux) | control of flux sweep over space mesh for one group |
| plane | (group) | control of flux sweep over one plane |
| output | (dotx) | output summaries and file writing; calls respns, refog |

## H.2 Special-Purpose Support Routines

### H.2.1 Routines Specific to TORT

| | | |
|---|---|---|
| cmsclr | (flux) | partial-current acceleration of fluxes |
| csol | (cmsclr) | control of acceleration matrix solution |
| csoler | (csol) | acceleration matrix convergence calculation |
| csolswp | (csol) | acceleration matrix iteration sweep |
| cmsumr | (group) | boundary flow calculation |
| cyyaa | (cyyac) | calculates AP flux correction |
| cyyac | (flux) | calculates AP acceleration factors |
| cyyaw | (cyyax) | calculates weights for AP acceleration |
| cyyax | (row) | interface for cyyaw |
| cyyaz | (cyyac) | calculates AP corrected flux |
| geom | (inpa) | space mesh setup and checking |
| inchk | (input) | input data checking |
| iocyl | (inpa) | disk file requirement calculation |
| macmx | (sorsum) | cross section file preparation and editing |
| map2 | (mapx) | zone map editing |
| mapr | (inpa) | zone map editing |

| | | |
|---|---|---|
| mapx | (inpa) | zone map array preparation |
| meshi | (group) | i-boundary spatial flux remapping |
| meshj | (plane) | j-boundary spatial flux remapping |
| meshk | (group) | k-boundary spatial flux remapping |
| ndx | (cyyaa) | cell indexing |
| pcon | (quad) | discrete direction/spherical harmonics moment coupler |
| quad | (inpa) | directional quadrature set processing |
| rebal | (flux) | acceleration factor application |
| refog | (output) | scalar flux output and edit |
| respns | (output) | response summary preparation and edit |
| row | (plane) | controls flux calculation for one row of the space mesh |
| rowlc | (row) | flux sweep of a row; scalar characteristic |
| rown | (row) | " ; standard scalar nodal |
| rownv | (row) | " ; standard vector nodal |
| rowy | (row) | source & moment calculations for y-series routines |
| rowyns | (rowy) | standard scalar nodal with AP capability |
| rowyu | (rowy) | unsafe (but fast) nodal |
| rowys | (rowy) | safe alternate scalar nodal |
| rowysv | (rowy) | safe alternate vector nodal |
| rowmp | (row) | flux sweep of a row; Octant Parallel multitasking on UNICOS |
| rowdp | (row) | flux sweep of a row; Direction Parallel multitasking on UNICOS |
| rownvp | (rowmp, rowdp) | flux sweep of a row for one discrete ordinate |
| sbsfix | (group) | boundary source implementation |
| sbsrin | (inpa) | boundary source file preparation |
| sdisin | (inpa) | distributed source file preparation |
| sflxin | (inpa) | flux moment file preparation |
| snplx | (sorx) | construction of directional source from moments |
| sorsor | (fscon) | source acceleration factor application |
| sorx | (plane) | negative source removal |
| wdflx | (sflxin) | input of flxmom- or varscl-formatted flux file |
| wrdir | (plane) | output of dirraw-formatted directional flux file |
| wrflx | (work,respns) | output of flxmom-formatted flux moment file |
| wrscl | (refog) | output of varscl-formatted scalar flux file |
| xmix | (cyyaa) | preconditioner mixing |

### H.2.2 Routines Shared with DORT

| | | |
|---|---|---|
| rowb | (row) | row flux calculation; scalar weighted difference |
| rowc | (row) | row flux calculation; vector weighted difference |
| rowd | (rowc) | dummy for cal-language theta-weighted routine |
| vario | --- | scratch file coordinator |

## H.3 General-Purpose Support Routines (Shared With DORT)

| | | |
|---|---|---|
| blkio | (vario) | controls random file access |
| bufio | (wrflx) | moves arrays to/from a single i/o record |
| cinc | --- | sets integer array to even-spaced values |
| clearx | --- | sets real data array to 0. |
| cmpri | (inpa) | compares integer arrays |
| cmultx | --- | multiplies real array by a constant |

| | | |
|---|---|---|
| csetx/cseti | --- | sets real/integer array to a constant |
| csuma | (geom,geos) | sums absolute value of real array |
| csumr | (input) | sums real array |
| cvrs | (input) | reverses columns or rows of real array |
| | | |
| edit | --- | prints irregular 1-, 2-d real/integer array |
| erro | --- | prints error message; records error severity |
| ffread | (fidas) | parses input stream record format for fidas |
| fidas | (fidos) | interprets input stream format |
| fidos | (loco,input) | controls fidas |
| | | |
| maxi | (input,inp) | finds maximum of integer array if length.gt.0 |
| minsa | (inchk) | finds max/min of integer array if length.gt.0 |
| msum | (input) | finds sum of integer array if length.gt.0 |
| msumd | (input) | finds indirect-address sum of integer array |
| | | |
| ndxr | (inpa) | calculates pivot arrays |
| nnzro | (input) | finds number of non-zero items in integer array |
| pack | --- | packs/unpacks strings of real data |
| | | |
| seqio | --- | controls sequential file access |
| skpedi | --- | gives quick diagnostic edit in several formats |
| timex/timset | --- | prints accumulated time usage |
| timsum/timon/timoff | --- | collects cpu time use by major category |
| wandr | (respns) | i/o of formatted/unformatted data file |
| wot10 | --- | prints 10-column edit of real/integer array |
| wot4 | --- | 1- to 4-d irregular real/integer array edit with header |
| wotv | --- | 1- to 4-d irregular real/integer array edit with header |

## H.4 Basic Unix Interface Packages

### H.4.1 Memory Allocation (CUNALOC)

| | | |
|---|---|---|
| dlocal | (dot,input) | keeps memory records, calls memorx |

### H.4.2 Support Library (CUNIO)

| | | |
|---|---|---|
| dopc | (vario,blkio,seqio) | opens, closes, destroys files |
| dred/drit | (blkio) | i/o's random-access files |
| fread/frit | (dred/drit) | short-list fortran direct-access i/o |
| cred/crit | (vario) | moves data from/to user direct-access buffer |
| | | |
| cmovx | --- | up-or-down word move with overlap protection |
| cmvbt | --- | byte move |
| cmpbt | (sorsum) | byte comparison |
| cright | (not used) | right adjustment of byte string |
| reed/rite | (seqio) | short-list fortran sequential i/o |
| | | |
| timez | --- | date/time/user services |
| header | (main) | performs program initialization |
| tailer/tailex/tailem | --- | timing and job wrapup |
| tailpr | (exitx) | prints summary of cpu & system time |
| exitx | (erro) | traceback and/or problem termination |
| minaf/maxaf | (sorx) | min/max of an integer array |

| | | |
|---|---|---|
| minmx | " | min and max of an integer array |
| aminaf/amaxaf | " | min/max of a real array |
| aminmx | " | min and max of a real array |
| seek | (not used) | bold venture interface unit allocation |

H.5 Cray Interface Package

H.5.1 Memory Allocation (CRAALOC; Overrides CUNALOC)

| | | |
|---|---|---|
| memorx | (dlocal) | calls system hpalloc, etc. |

H.5.2 Support Library (CRAIO; Overrides CUNIO)

| | | |
|---|---|---|
| dumcos/flush | --- | dummy entries |
| timget | (timez) | calls clock, date; returns formatted time/date |
| destry | " | destroys an external disk file |

H.5.3 C Language (CRACCC)

| | | |
|---|---|---|
| csecond | (timez) | cpu+system time, seconds |
| userdata | " | returns userid, group, process number |

H.5.4 Cray CAL Language (CRAOPAL overrides routines of the same name)

| | | |
|---|---|---|
| locf | (not used) | word address locator |
| rowd | (rowc) | short-vector theta-weighted difference |
| rowi | (not used) | 2-d zero-weighted flux calculation for a row |

H.5.5 Cray CAL2 Language (CRACAL2 overrides routines of the same name)

| | | |
|---|---|---|
| locf | (not used) | word address locator |
| rowd | (rowc) | short-vector theta-weighted difference |
| rowi | (not used) | 2-d zero-weighted flux calculation for a row |

H.6 IBM Interface Package

H.6.1 Memory Allocation (CIBALOC, overrides CUNALOC)

| | | |
|---|---|---|
| memorx | (dlocal) | calls system routines malloc, free |

H.6.2 Support Library (CIBIO overrides CUNIO)

| | | |
|---|---|---|
| dumibm/flush | --- | dummy entries |
| timef | (timez) | calls e_clock, returns time-of-day, msec |
| second | " | calls mclock, returns cpu time, seconds |
| csecond | " | calls etime, returns cpu+system time, seconds |
| timget | " | calls fdate; returns formatted time/date |
| userdata | " | calls getlog, getpid; returns userid, group, process |

H.6.3 C Language (CIBCCC)

| | | |
|---|---|---|
| tdate | (not used) | time/date character string |
| e_clock | (timef) | time-of-day, hundredths of seconds |

H.7 Sun Interface Package

H.7.1 Memory Allocation (CSNALOC overrides CUNALOC)

| | | |
|---|---|---|
| memorx | (dlocal) | calls system malloc, free |

H.7.2 Support Library (CSNIO overrides CUNIO)

| | | |
|---|---|---|
| dumibm/flush | --- | dummy entries |
| timef | (timez) | calls time, returns time-of-day, msec |
| second | " | calls etime, returns cpu time, seconds |
| csecond | " | returns cpu+system time, seconds, as 0. |
| timget | " | calls tdate; returns formatted time/date |
| userdata | " | returns userid, group, process as blank |

H.7.3 C Language (CSNCCC)

| | | |
|---|---|---|
| tdate | (timeget) | gets time/date string |

H.8 DEC Interface Package

H.8.1 Memory Allocation (CDCALOC overrides CUNALOC)

| | | |
|---|---|---|
| memorx | (dlocal) | calls malloc, free |

H.8.2 Support Library (CSNIO overrides CUNIO)

Note: The SUN support library is used here. Language flags change a call to tdate into a call to fdate.

H.8.3 C Language (CSNCCC)

Note: The SUN C-language package is used here without change.

H.9 HP Interface Package

H.9.1 Memory Allocation (CSNALOC overrides CUNALOC)

| | | |
|---|---|---|
| memorx | (dlocal) | calls malloc, free |

H.9.2 Support Library (CSNIO overrides CUNIO)

Note: The SUN support package is used here without change.

H.9.3 C Language (CHPCCC)

| | | |
|---|---|---|
| etime | (second) | gets cpu time, seconds |

H.10 Generic Interface Package (* Not Presently Used *)

H.10.1 Memory Allocation (CXXALOC overrides CUNALOC)

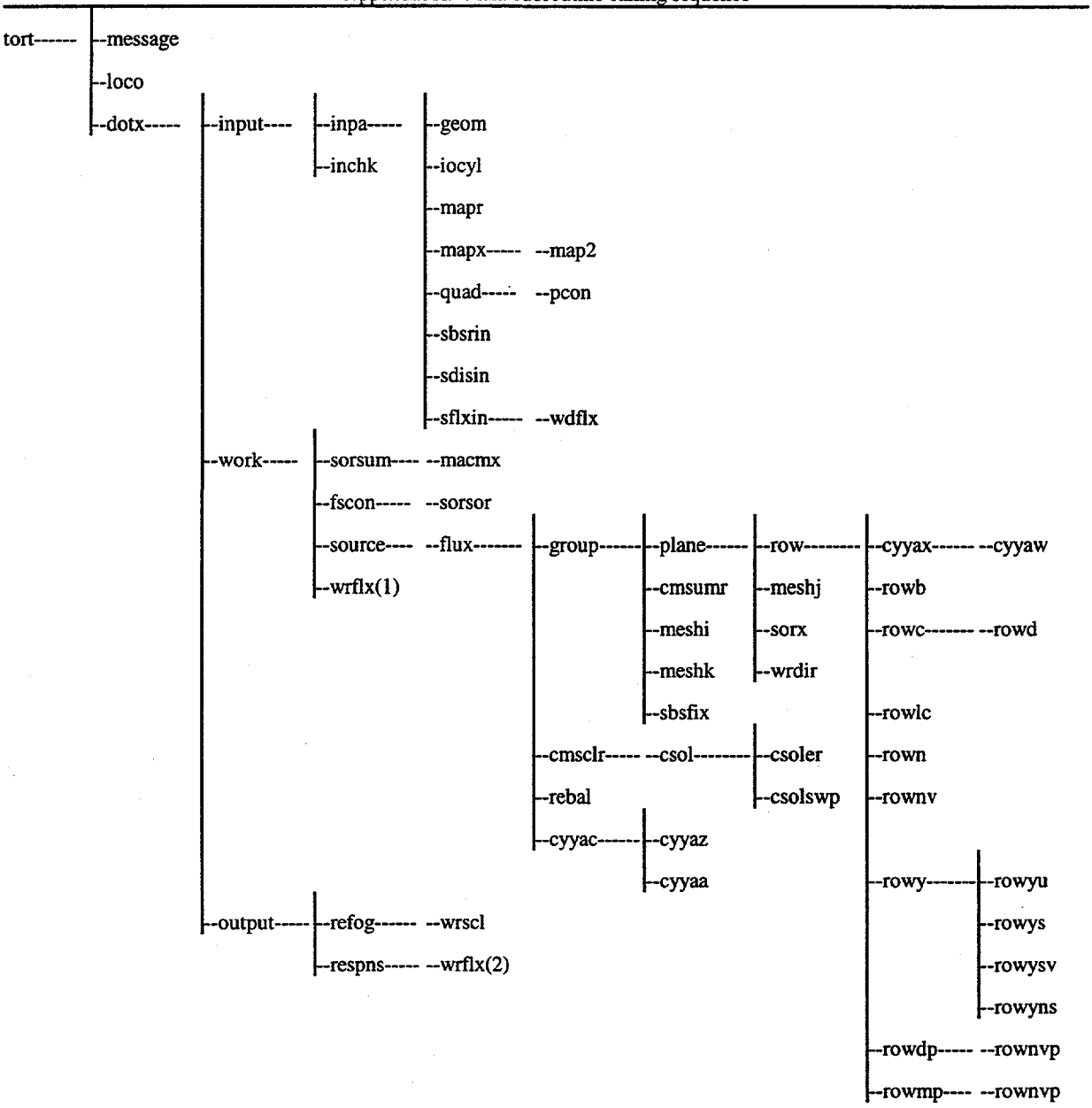
| | | |
|---|---|---|
| memorx | (dlocal) | establishes large fixed-dimension container space |

H.10.2 Support Library (CXXIO overrides CUNIO)

| | | |
|---|---|---|
| dumibm/flush/trbk | --- | dummy entries |
| timef | (timez) | returns time-of-day, msec, as 0 |
| second | " | returns cpu time, seconds, as 0 |
| csecond | " | returns cpu+system time, seconds, as 0. |
| timget | " | returns formatted time/date as null |
| userdata | " | returns userid, group, process as blank |

H.10.3 C Language (CXXCCC)

| | | |
|---|---|---|
| malloc | (not used) | "system-independent" space allocation |




Note: wrflx and rownvp are used in two places.

# APPENDIX I. DEMONSTRATION PROBLEM SET

In some cases, problems are inadequately converged or use inadequate mesh or quadrature in order to illustrate a particular feature or to make a particular point without unnecessary expense. In a few cases, undocumented options are used, and the user should not attempt to make general use of such features.

1. The first set of problems uses 2 related cross section libraries. The first, a 1-group p1 set, features a pure scatterer, a pure absorber, a "half-scatterer" with half scattering and half absorption, and a near-void. All except the near-void have an average scattering cosine of 1/3. Since Oak Ridge cross sections have sigma-1 equal to $(2\ell+1)$ times the average scattering cosine, sigma-1 and sigma-0 are equal for the selfscatter positions in those sets. The near-void has an average cosine near unity.

   The second library, a 2-group p1 set, is placed onto unit 4 rather than the default unit, 8. With the exception of the half-scatterer, p0 data for the two groups are identical to the 1-group library. The second group has symmetrical scattering; i.e., sigma-1 of 0. The half-scatterer has sigma-total for its first group split evenly between absorption and scattering to the second group. The second group sigma-total is split evenly between absorption and fission. Two product particles are produced per fissile capture.

   The first tort problem is a small, 1-cell core of the half-scattering material in one corner of a 3x3x3 XYZ cube of full-scattering material. Reflection is used at the explicit boundaries, i.e., the left, inside, and bottom. S2 directional quadrature is used with the 2-group p1 cross section library. The problem is started with an arbitrary fixed source guess. The effect of the source guess disappears as the iteration proceeds. The flux output is written on unit 1. The problem is forced to hold only 1 group of data in memory at a time.

   The problem is then restarted from the flux guess. It is necessary to supply an estimate of k-effective in order to normalize the flux guess accurately for a smooth restart. The problem is forced to split the data storage for a group into 2 blocks.

   The next problem begins the use of the 1-group library with a homogeneous 3x3 XZ mesh of scattering material. The source is in the first mesh cell. This problem will be compared with a DORT problem later. Following this, a 4x4 scattering source region is placed at the corner of an absorbing 8x8 xy grid. This problem is also for comparison with DORT. In the last TORT problem, the 8x8 xy problem is extended to XYZ geometry.

   If the dos driver is used to control the solution, execution will continue with 2 DORT problems. The first compares accurately with the 3x3 XZ TORT problem, while the second compares with the 8x8 xy problem.

2. This problem set uses 3-group p3 cross sections borrowed from the ANISN sample problem 1. They were originally taken from a small space reactor problem. The 4 zones of the original reactor design are detailed: core, interface zone, radial reflector, and axial reflector. The dimensions of the problem have been reduced, however.

The first TORT problem is an S2P1 search for k-effective. In the second problem, chi is set to 0.99/k, thus making the system near-critical. Then, a uniform fixed source in the core is added, and the converged multiplication of that source is sought. Since the problem is small, the global extrapolation procedure is satisfactory in this case.

Finally, the same problems are solved with DORT. Both problems reproduce the corresponding TORT results well. Since multiplication problems are very sensitive to small procedural differences, the agreement in M is not as precise as the agreement in k, however. For example, the smallest relative difference in k that can be printed in this problem is 5e-6. Such a systematic difference would result in a 5e-4 difference in M values, and this of the same order of magnitude as the difference in M actually observed.

3. This problem set uses a 14-group library taken from a problem used to demonstrate the irregular-mesh features of DOT 4 at the *1977 Fifth International Shielding Conference* in Knoxville. A concrete cylinder encased in a steel shell is driven by a source at the top in RZ geometry or at the outside surface in r-theta geometry. Only the first 3 energy groups are solved. The cross sections are P3, but the solution is conducted in P1, since the P3 basis functions are not orthogonal over the discrete S4 direction set. Serious production problems should use at least S6 with P3 cross sections.

   Problem 3.1 uses RZ geometry to demonstrate the most efficient weighted difference flux routines on the computer being used, and then 3.2 uses the alternate routines. The two procedures give the same result. Problem 3.3 solves the same problem with r-theta-z geometry, reflected at both theta boundaries. The results match very closely.

   In problem 3.4, the geometry is changed to r-theta, and the source is at the outer boundary, i.e., at the maximum theta position. In problem 3.5, the same problem is solved in r-theta-z geometry, with reflection at both top and bottom. The flux is uniform in z and in agreement with the r-theta problem. It may be noted that a large z mesh was used. If a small mesh is used, the agreement is not so favorable. This may be due to limitatns of the weighted difference procedure, although that has not been proven satisfactorily.

   In problem 3.6, the r-z problem of 3.1 is solved with DORT, proving the correctness of the TORT result. Similarly, problem 3.7 solves the r-theta problem of 3.4, confirming the TORT result.

4. The library for this problem is a 13-neutron, 7-gamma collection that has seen considerable use in analyzing transport into large buildings at the preliminary level. The building, itself, is a "metric doghouse" built of simple concrete slabs. The problem set is intended to provide a low-cost test of several of the numerical formulations and restart provisions. Only the first 3 energy groups are solved. S2P1 is used in all solutions.

   Case 4.1 uses the standard linear nodal routine and saves the random access files on units 91 and 94 for reuse. the first group is converged completely, but only 3 iterations each are allowed for groups 2 and 3.

Case 4.2 reuses the files on units 91 and 94 as input guesses. INGRPS specifies that only 2 groups of the input are to be used, the rest set to 0.0 by default. The value of NIFCNV causes iterations in the first group, which had previously converged, to be skipped entirely. It also causes the acceleration damping of the second group to continue from the previous level, promoting a smooth convergence process. The output indicates that the flux for the third group was ignored.

The remaining cases test the remaining higher values of the mode parameter.

5. Using the same cross section set, the "buzz house" is solved next. This building has two rooms divided by a concrete center partition, an internal pillar, two windows on each side, and a single window on the end facing the source. It represents a building actually constructed in 1984 and used in experiments designed to test the accuracy of TORT. Details of the experiment can be found in ORNL/TM-9528.

   This problem is more complex than many production TORT problems in that it uses multiple sources and has provisions to rearrange the geometry to several configurations. As the problem is set up, it will run only the first energy group as a test. By disabling the line containing the last 64$$ array and all lines between the first "stop" line and the "t" preceding it, inclusive, the problem will run the full set of energy groups.

   The geometric mesh is arranged with the z axis toward the front of the building and the y axis upward. By manipulating the 8$$ arrays, the pillar can be relocated, the partition can be included or removed, and the end window can be opened or closed to reflect configuration changes. As a matter of economy, the 6" internal wall is smeared into a 12" space and assigned a density factor of 0.5. The air inside the building is assigned full convergence importance, while everything else is assigned 0.1. A total of 34 key locations are selected for special monitoring. The iteration limits allow extra iterations for the thermal neutron group, since it tends to be difficult to converge.

   It happens that the acceleration formulation used here is not entirely effective if the internal boundary sources are used. Accordingly, the acceleration is disabled in several groups for which this defect has been found to cause trouble. This weakness in the formulation is not complicated, and a repair will be made eventually.

   A choice of several source spectra is provided, and responses corresponding to the installed instrumentation follow. Finally, the directionality, spatial shape, and spectrum of the external boundary source is described, followed by 3 internal boundary sources corresponding to the front wall, pillar plane, and back wall surfaces.

   Although this problem does not represent the final analysis of the subject experiment, it is illustrative of the general approach and the capabilities of the code.

6. Still with the same cross section set, a very large building solution is demonstrated. This geometry represents one of the reinforced concrete buildings subjected to nuclear weapon attack during World War II. The geometry details walls, windows, and support members individually. The source is taken to be uniform over the right, inside, and top surfaces. Only 1 group of the

20-group problem is normally solved as a test problem, although it is easy to activate the full calculation. The total flux at 10 key locations is monitored to indicate the results of the calculation.

In the actual production study, it was necessary to add a basement, internal walls, and more structural accuracy, making the geometric description much longer. The boundary source was obtained by interpolation of the flux from a 2-d air/ground calculation. Several source components were treated separately, requiring a number of individual runs. A 69-group cross section library was used. Some of the calculations were done with s10 directional quadrature, rather than the s6 used here, and some used the linear nodal flux formulation. thus, the production studies were much more ambitious in scope than this example.

7. Still with the 20-group library, a set of problems demonstrating the use of the discontinuous mesh feature on various model buildings is solved. The first model is a concrete-air construction with (4,2)*1*3 XYZ mesh, i.e. with either 4 or 2 mesh intervals in the x direction, depending upon y and z, with 1 interval in the y direction, and 3 intervals in the z direction. The problem is a 2-group P1 nodal solution. More directions would be needed to produce accurate results, but the set used is adequate for this test. Then, the problem is solved as a 1*(4,2)*3 XYZ mesh and again as a (4,2)*3*1 XYZ problem, providing a severe test of the remeshing facilities.

   Next, this same model is solved as a 2-group P1 RZ weighted difference problem with discontinuous mesh and distributed source for comparison with DORT, proving that these various features can be used together.

   A quarter-symmetry version of the buzz house used in an earlier problem set is next solved in XZ geometry using a 2-group P1 weighted difference calculation with a boundary source. Results from this problem will also be compared with DORT. It uses negfix.gt.0 for negative removal without disturbing balance. It can also be solved using the nodal procedure by setting mode=2, ingeom=0, and theta=0.

   In the last TORT test, the quarter-symmetry buzz house is solved in XYZ geometry with multiple mesh sets used in both x and y dimensions. A larger directional quadrature set is also used, a set with low-weight "carrier directions" that allow the boundary source to be concentrated near the z axis, and p3 scattering is used. The nodal flux procedure is used, and negfix=2. This problem can also be solved with weighted difference, but the convergence levels off before reaching 1.0e-3 and locks into a limit cycle. This is apparently due to the basic nature of the SORX procedure used when negfix.gt.0, and it is not exclusively a problem with weighted difference.

   In two DORT tests, the RZ and XZ cases described above are solved by DORT, producing essentially the same result obtained by TORT in each case.

8. The cross sections are a variant of those used for problem set 1, differing primarily in that mubar for material 3 on output unit 4 is 1/3 in both energy groups, but the scatter from group 1 to group 2 is isotropic.

Problem 8.1 describes a metric doghouse similar to that of problem 4. Weighted difference is used, and key responses at several points are edited. Problem 8.2 repeats this problem with a discontinuous space mesh installed, but only one i-set and one j-set are used. The results are unaffected. In problem 8.3, the full discontinuous mesh is used to reduce the number of space cells from 45 to 19. All of the results are affected, of course, but only the last key flux has what must be regarded as a large change. The integrals remain close to the previous values. An experiment not shown indicated that the change in the key fluxes could be reduced to a nominal amount by using the nodal procedure. Nodal would not be acceptable in the p0 problem to be solved next, however.

In problem 8.4, the second group is restarted using S2 P0 rather than the S4 P1 used in all of the above problems. Modest changes in the results are noted. Finally, problem 8.5 demonstrates that the same result could be obtained in one step using the variable quadrature and scattering expansion features.

It should be noted that the success of this comparison depends upon the fact that the downscatter cross section was 0. In problem 8.4, since P0 was specified for all energy groups, the downscatter was calculated in P0. In 8.5, however, the first group is solved in P1, and so the scattering source into group 2 is, in principle, P1. Since the corresponding cross section has been set to 0, however, the results match as desired.

## INTERNAL DISTRIBUTION

1-5. Y. Y. Azmy
6. J. M. Barnes
7. E. D. Blakeman
8. S. M. Bowman
9. B. L. Broadhead
10. J. A. Bucholz
11. L. A. Charlton
12. R. L. Childs
13. M. D. DeHart
14. F. C. Difilippo
15. J. D. Drischler
16. M. B. Emmett
17. S. E. Fisher
18. T. A. Gabriel
19. F. X. Gallmeier
20. J. C. Gehin
21-25. D. T. Ingersoll
26. J. O. Johnson
27. M. A. Kuliasha
28. L. C. Leal
29. R. W. Lee
30. R. A. Lillie
31. J. B. Manneschmidt
32. G. S. McNeilly
33. J. V. Pace, III
34. C. V. Parks
35. L. M. Petrie
36. R. T. Primm, III
37. I. Remec
38. J. P. Renier
39. R. W. Roussin
40. R. T. Santoro
41-45. D. B. Simpson
46. C. O. Slater
47. J. S. Tang
48. R. M. Westfall
49-53. B. A. Worley
54. J. J. Yugo
55. CPED Reports Office
56. Laboratory Records Dept.
57. Laboratory Records, ORNL-RC
58. Central Research Library Document Reference Section
59. ORNL Patent Section

## EXTERNAL DISTRIBUTION

60. M. Bailey, U.S. Nuclear Regulatory Commission, Office of Nuclear Material Safeguards and Safety, IMNS STSB, MS 8 F5, Washington, D.C. 20555.
61. W. A. Rhoades, 110 Netherlands Road, Oak Ridge, TN 37830.
62. Knolls Atomic Power Lab, P.O. Box 1072, Room D2-114, Schenectady, New York 12301.
63. Dr. Allen Barnett, Knolls Atomic Power Lab, P.O. Box 1072, Room D2-114, Schenectady, New York 12301.
64. Charles Burre, Knolls Atomic Power Lab, P.O. Box 1072, Room D2-114, Schenectady, New York 12301.
65. Art Rubin, Knolls Atomic Power Lab, P.O. Box 1072, Room D2-114, Schenectady, New York 12301.
66. Dr. Enrico Sartori, OECD NEA Data Bank, Le Seine Saint-German, 12 Blvd. Des Iles, F-92130 Issy-les-Moulineaux.
67. Colonel Mark E. Byers, DP-22, A-301/GTN, Germantown, MD 20874.
68. Major David B. Myers, Defense Nuclear Agency, ATTN: SPWE, 6801 Telegraph Road, Alexandria, Virginia 22310-3398.
69. Al Taboada, U.S. NRC, Two White Flint N., Room 10027, 11545 Rockville Pike, N. Bethesda, MD 20852-2378.
70. Harry Alter, Office of International Nuclear Safety, NE-30, A-162/GTN, 19901 Germantown Road, Germantown, MD 20874.

71. Dr. Charles M. Ward (IANG-TMT), U.S. Army National Ground Intelligence Center, 220 7th Street, NE, Charlottesville, VA 22902-5396.
72. Director, The Defense Special Weapons Agency, ATTN: RAEM (Robert A. Kehlett), 6801 Telegraph Road, Alexandria, Virginia 22310-3398.
73. Dr. I. K. Abu-Shumays, Bettis Atomic Power Laboratory, Box 79, West Mifflin, PA 15122.
74. Carol Yehnert, Bettis Atomic Power Laboratory, Box 79, West Mifflin, PA 15122.