LA-10049-M
Manual

c.1

User's Guide for TWODANT: A Code Package for Two-Dimensional, Diffusion-Accelerated, Neutral-Particle Transport

Los Alamos

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

This work was supported by the US Department of Energy, Office of Breeder Reactor Technology.

LA-10049-M
Manual

UC-32
Issued: March 1984

User's Guide for TWODANT: A Code Package for Two-Dimensional, Diffusion-Accelerated, Neutral-Particle Transport

Ray E. Alcouffe 5956
Forrest W. Brinkley 4680
Duane R. Marr 5935
R. Douglas O'Dell 6551

Los Alamos

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

CONTENTS

USER'S GUIDE FOR TWODANT:

A CODE PACKAGE FOR TWO-DIMENSIONAL, DIFFUSION-ACCELERATED, NEUTRAL-PARTICLE TRANSPORT

by

Ray E. Alcouffe, Forrest W. Brinkley,
Duane R. Marr, and R. Douglas O'Dell

ABSTRACT

1. Program identification: TWODANT

2. Computer for which Program is designed: CDC-7600, but the program has been implemented and run on the IBM-370/190 and CRAY-1 computers.

3. Function: TWODANT solves the two-dimensional multigroup transport equation in x,y and r,z geometries. Both regular and adjoint, inhomogeneous (fixed source) and homogeneous (k-effective) problems subject to vacuum, reflective, periodic, or white boundary conditions are solved. General anisotropic scattering is allowed and anisotropic inhomogeneous sources are permitted.

4. Method of Solution: TWODANT numerically solves the two-dimensional, multigroup form of the neutral-particle, steady-state Boltzmann transport equation. The discrete-ordinates form of approximation is used for treating the angular variation of the particle distribution and the diamond-difference scheme is used for space-angle discretization. Negative fluxes are eliminated by a local set-to-zero-and-correct algorithm. A standard inner (within-group) iteration, outer (energy-group-dependent source) iteration technique is used. Both inner and outer iterations are accelerated using the diffusion synthetic acceleration method. The diffusion solver uses the multigrid method.

5. Restrictions: The code is thoroughly variably dimensioned with a flexible, sophisticated data management and transfer capability. The code is designed for a three-level hierarchy of data storage: a small, fast core central memory (SCM), a fast-access peripheral large core memory (LCM), and random-access peripheral storage. (For computing systems based on a two-level hierarchy of data storage - a large fast core and random-access peripheral storage - a portion of fast core is designated as a simulated LCM to mimic the three-level hierarchy). Random-access storage is used only if LCM (or simulated LCM) storage requirements are exceeded. Normally, an SCM of about 25 000 words of storage and an LCM (or simulated LCM) of a few hundred thousand words or less storage are sufficient to eliminate the need for using random-access storage.

6. Running Time: Running time is directly related to problem size and to central processor and data transfer speeds. On the CRAY-1, a four group, adjoint calculation of the eigenvalue of an R-Z model of the Fast Test Reactor (FTR) took 15 seconds. The calculation used transport corrected P0 cross sections, an S4 angular quadrature, and a 31 by 68 spatial mesh.

7. Unusual Features of the Program: The TWODANT code package is modularly structured in a form that separates the input and the output (edit) functions from the main calculational (solver) section of the code. The code makes use of binary, sequential data files, called interface files, to transmit data between modules and submodules. Standard interface files whose specifications have been defined by the Reactor Physics Committee on Computer Code Coordination are accepted, used, and created by the code. A totally new free-field card-image input capability is provided for the user. The code provides the user with considerable flexibility in using both card-image or sequential file input and also in controlling the execution of both modules and submodules. Separate versions of the code exist for short-word and long-word machines.

8. Programming Languages: The program is written in standard FORTRAN-IV language.

9. Machine Requirements: Six Input/Output units and up to 14 interface units are required. The number of interface units needed is problem dependent. Typically, 10 such units are used. For CDC-7600 computers a 50 000 word small core (SCM) and a large core memory (LCM) are required. For computers with only a single fast core, the fast core size must be sufficiently large to permit partitioning into an SCM and simulated LCM. Random-access auxiliary storage may occasionally be required if LCM (or simulated LCM) storage is insufficient for the problem being executed.

10. Material Available: Source deck (about 45 000 card images), sample problems, this manual, and the ONEDANT manual(Ref. 2).

I. INTRODUCTION

The TWODANT code package is a modular computer program designed to solve the two-dimensional, time-independent, multigroup discrete-ordinates form of the Boltzmann transport equation. TWODANT uses the same modular construction as the ONEDANT code. This modular construction separates the input processing, the transport equation solving, and the postprocessing, or edit functions, into distinct, independently executable code modules, the INPUT, SOLVER, and EDIT modules, respectively. These modules are connected to one another solely by means of binary interface files. The INPUT module and, to a lesser degree, the EDIT module are general in nature and are designed to be standardized modules. With these modules, different new production codes can be created simply by developing different SOLVER modules that can be "plugged in" to the standardized INPUT and EDIT modules.

The TWODANT code is simply the ONEDANT package with the one-dimensional SOLVER module replaced with a two-dimensional SOLVER module. As such, large portions of the ONEDANT manual apply to TWODANT as well. This TWODANT user's guide follows the ONEDANT manual form even to the point of having the same chapter and section numbering. Thus, Chapter IV, section C contains the full input specifications, in either manual. Frequently, for sections which apply equally to both codes, a very brief discussion will be given here and the user will be referred to the ONEDANT manual for more details. This is only done for sections that are of background interest or for sections that go into great detail. Once the general scheme of things is known, the user should only very infrequently need to access the ONEDANT manual. Thus, this TWODANT user's guide is intended to be complete in the sense that all the input arrays are described along with everything else needed to make most runs.

Some of the major features included in the TWODANT package are:

(1) a totally new, free-field format card-image input capability designed with the user in mind,

(2) highly sophisticated, standardized, data- and file-management techniques as defined and developed by the Committee on Computer Coordination (CCCC) and described in Ref. 1; both sequential file and random-access file handling techniques are used,

(3) the use of a diffusion synthetic acceleration scheme to accelerate the iterative process in the SOLVER module,

(4) direct (forward) or adjoint calculational capability,

(5) x,y and r,z geometry options,

(6) arbitrary anisotropic scattering order,

(7) vacuum, reflective, periodic, and white boundary condition options,

(8) inhomogeneous (fixed) source or k-effective calculation options,

(9) "diamond-differencing" for solution of the transport equation,

(10) a new diffusion solver that uses the multigrid method,

(11) user flexibility in using either card-image or sequential file input,

(12) user flexibility in controlling the execution of both modules and submodules, and

(13) extensive, user-oriented error diagnostics.

TWODANT is a large, very flexible code package. Great effort has been devoted to making the code highly user-oriented. Simple problems can be easily run and many of the code options can be ignored by the casual user. At the same time numerous options for selective and sophisticated executions are available to the more advanced user. In all cases redundancy of input has been minimized, and default values for many input parameters are provided. The code is designed to be "intelligent" and to do much of the work for the user. The input is designed to be meaningful, easily understood, easily verified, and easy to change. The printed output is well documented with liberal use of descriptive comments and headings. In short, TWODANT was designed to be fun to use.

Chapter II of this manual provides the user with a brief overview of the code package. Included are sections on programming practices and standards, code package structure, and functional descriptions of the three principal modules comprising the package. All this information applies to ONEDANT as well, and more complete discussions of each section can be found in the ONEDANT manual.

Chapter III presents the free-field format rules for the card-image input. The more esoteric forms of input that both ONEDANT and TWODANT support are not described here and the user is referred to the ONEDANT manual for a discussion of those items.

Chapter IV provides the card-image input specifications for TWODANT. First is given an overview of the specification of input including descriptive examples. Next is a "mini-manual" on which are listed all the available input arrays arranged by input block. This mini-manual is very useful to the user in organizing his input. For the more experienced user, the mini-manual is frequently all that is needed for him to specify his input. Following the mini-manual is a moderately detailed description of all the input parameters and arrays.

Chapters II, III, and IV should be read by all first-time users of TWODANT.

Chapters V through IX of the ONEDANT manual largely apply to TWODANT as well, and are not repeated in this guide. Those chapters are briefly described below.

Chapter V provides the interested user with details related to the input for ONEDANT. Included is a brief development of the multigroup, discrete-ordinates form of the diamond-differenced Boltzmann transport equation. This section is followed by numerous sections providing specific detailed information needed by the user to fully understand some of the input options and input arrays. The chapter supplements the information presented in Chapter IV.

Chapter VI gives details related to the actual execution of the SOLVER module. Described are the iteration strategy, convergence criteria, termination criteria for the iterative loops, and the iteration monitor print provided by the code.

Chapter VII is devoted to details related to the EDIT module of the code. Both input and execution control options for this module are described in detail. This chapter supplements information pertaining to the EDIT module provided in Chapter IV.

In Chapter VIII is a discussion of some of the more sophisticated options available to the advanced user for controlling the execution of modules and submodules in ONEDANT.

Chapter IX presents a discussion of the error diagnostics available in ONEDANT. Several examples of errors and the resulting error messages are provided for the user.

The information in four appendices is germane to the running of TWODANT. Appendix A of the ONEDANT manual provides the file descriptions for the code dependent, binary, sequential interface files generated by and used in the ONEDANT code package. As these are identical to the files used by TWODANT, that appendix is not repeated here. File descriptions for the CCCC standard interface files are also not provided, but can be found in Ref. 1. Appendices B, C, and D are found in this user's guide. Appendix B provides a sample TWODANT problem for the user. Appendix C shows how to access the code at Los Alamos and Appendix D shows the access at the National Magnetic Fusion Energy Computer Center at Livermore, California.

II. TWODANT OVERVIEW

The TWODANT code package is a computer program designed to solve the two-dimensional, multigroup, discrete-ordinates form of the neutral particle Boltzmann transport equation. It was developed as a modular code package consisting of three modules: an INPUT module, a SOLVER module, and an EDIT module.

In this chapter is provided a brief discussion of the general programming practices and standards used in the code package, a description of the code structure, and overviews of the three modules comprising the package.

A. Programming Practices and Standards

In general, the programming standards and practices recommended in Ref. 1 from the Committee on Computer Code Coordination (CCCC) have been followed throughout the development of TWODANT. By following these practices and standards, problems associated with exporting and implementing the code in different computing environments and at different computing installations are minimized. The programming practices and standards are also described in more detail in the ONEDANT manual (Ref. 2) and the user is referred thereto.

B. TWODANT Code Package Structure

The TWODANT code package consists of three major functionally independent modules: an INPUT module, a SOLVER module, and an EDIT module. These modules are linked by means of binary interface files. The INPUT module processes any and all input specifications and data and, if required, generates the binary files for use by the SOLVER and/or EDIT modules. The SOLVER module performs the transport calculation and generates flux files for use by the EDIT module. The SOLVER module also generates other interface files for use by other codes or for subsequent calculations by the SOLVER module. The EDIT module performs cross-section and response function edits using the flux files from the SOLVER module.

The interface files accepted, used, and generated by the modules are described in detail in the ONEDANT manual. Also shown there is the relationship of each of the files to each of the modules, where it is used, where it is written, and so forth. That detail is not repeated here. The fact that interface files are used at all is not generally useful to the user unless he is making his runs in piecewise fashion.

A three-level overlay structure is used in TWODANT for implementing the modules. Such a structure involves the use of a main overlay together with primary and secondary overlays.

The main (or 0,0) overlay contains the main program routine, which controls the calling of the primary overlays, together with those service subroutines used by more than one primary overlay.

The first overlay constitutes the INPUT module. It is structured into the first primary (or 1,0) overlay plus twelve secondary overlays, each of which performs a unique function, usually the reading of a Block of input and the writing of one of the interface files. The INPUT module is also constructed in modular form and indeed may even be executed piecewise, if so desired. This module is identical to the INPUT module of ONEDANT.

The second overlay constitutes the SOLVER, or calculational, module. It consists of the second primary (or 7,0) overlay plus nine secondary overlays, each performing a part of the flux calculation.

The third overlay is the EDIT module. It currently consists of the third primary, or (3,0), overlay plus a single secondary overlay. This module is identical to the EDIT module of ONEDANT.

Normally all three of these modules will be executed in a given run although it is possible to run them individually given the necessary interface files. For a complete discussion of this latter technique, see the ONEDANT manual (Ref. 2).

A fourth overlay is also used in TWODANT. This overlay contains only the fourth primary, or (4,0), overlay with two subroutines. This fourth overlay provides highlights of the just-executed run as an aid to the user. These highlights are a printed summary of some of the pertinent facts, options, and decisions encountered during the run along with storage and run time information. This overlay is not considered to be a module in the sense of the first three overlays.

C. INPUT module

The INPUT module performs the necessary activities for processing all input data required for the execution of the SOLVER and/or EDIT modules. These activities include the reading of input data and the creation of binary interface files. The latter activity may require a certain degree of data processing. Each of these activities is discussed below.

In performing the reading-of-input data activity, the INPUT module accepts standard interface files (binary), code-dependent binary interface files, or card-images for its input. Input data can be provided in several different forms and many combinations of forms to provide a great deal of flexibility to the user. Chapter IV of this manual and Chapters IV, V, VI, and VII of the ONEDANT manual provide specific information and further details on the specification of the input data.

The second major activity in the INPUT module is the creation of binary interface files containing all input data. These files are subsequently used as the sole means of transmitting data to either the SOLVER or EDIT modules. The files emerging from the INPUT module take the form of either CCCC standard interface files or code-dependent interface files. In this file creation activity the INPUT module is called on to perform several types of tasks. As an example, the only form in which geometry related information emerges from the INPUT module is in the form of a GEODST standard interface binary file. If a user supplies geometry related input by means of card-image input, a particular submodule of the INPUT module reads this input, translates the data into a GEODST compatible form, and creates the resulting GEODST file. On the other hand, if the geometry related information is supplied by the user through an already existing GEODST file, the INPUT module is required to do nothing. In either case, the GEODST file will be available for the SOLVER module to use when doing the flux calculation. A second, more complex example of the function of the INPUT module involves the mixing of isotopes, or nuclides, to create Materials which are subsequently assigned to physical regions in the problem (called Zones) to define the macroscopic cross-section data for the Zones. For this example it will be assumed that the user selects card-image input as the form for the INPUT module. First, the isotope mixing specifications appropriate for the desired Materials are input via card-image. The INPUT module reads this data, translates the data, and creates the two standard interface files NDXSRF and ZNATDN. Assuming next that the isotope cross sections are provided by the user as a card-image library, another submodule of the INPUT module reads this library (in isotope ordered form) and also reads the just created NDXSRF and ZNATDN files. The mixing specifications provided by the latter files are applied to the isotopic cross-section data to generate Material cross sections; these mixed cross sections are then sorted into group order and written to the MACRXS code-dependent binary interface file. (A group ordered file named SNXEDT for use by the EDIT module is also created at this time but will not be considered in this example.) The MACRXS file becomes the sole source of cross-section data to the SOLVER module if the SOLVER calculation is to be a forward, or regular, calculation. If an adjoint calculation is to be performed by the SOLVER, yet another submodule of the INPUT module rereads the MACRXS file, performs the adjoint reversals on the cross sections, and creates the code-dependent binary file named ADJMAC containing the adjoint reversed Material cross sections for use by SOLVER.

D. SOLVER Module

The SOLVER module of TWODANT has the function of effecting numerical solutions of the two-dimension, multigroup form of the neutral particle steady-state Boltzmann transport equation. The discrete ordinates approximation is used for treating the angular variation of the particle distribution and the diamond difference scheme is used for spatial discretization.

In solving the transport equation numerically, an iterative procedure is used. This procedure involves two levels of iteration referred to as inner and outer iterations. The acceleration of these iterations is of crucial importance to transport codes in order to reduce the computation time involved. The TWODANT SOLVER module employs the diffusion synthetic acceleration method developed by Alcouffe (Ref. 3), an extremely effective method for accelerating the convergence of the iterations. Those unfamiliar with the inner/outer iteration process or the diffusion acceleration schemes are referred to the ONEDANT manual (Ref. 2) and/or to Ref. 3.

The diffusion solver, which does a major part of the calculational work, uses the multigrid method. This method is considered an improvement over the line successive overrelaxation (LSOR) method of solution common to most other two-dimensional diffusion solvers. It is faster converging for fine mesh problems and extends the domain of practical application to problems which would take unreasonable computation times to converge using LSOR. The multigrid method is described in Ref. 4.

E. EDIT Module

The function of the EDIT Module is to produce the printed edit-output selected by the user. Edit-output refers to information which is obtained from the data contained on one or more interface files but which generally requires manipulating or processing of the data. An example of the edit-output is a microscopic reaction rate distribution, i.e. the product of a microscopic cross section for a particular isotope or nuclide and the flux. In this example, the data from both a cross-section interface file and a scalar flux file are required to be recovered, multiplied, and the product printed.

The EDIT module is an essentially free standing module accepting only interface files as input and producing printed output. The required input files for execution of the EDIT module are the code-dependent binary interface file EDITIT and the standard interface files RTFLUX (or ATFLUX) and GEODST. Optional input files are the standard interface files NDXSRF and ZNATDN and the code-dependent files SNXEDT and ASGMAT. The code-dependent files are produced by the INPUT module.

III. FREE FIELD INPUT

There are four basic quantities in the free field input used in TWODANT; they are ARRAY, DATA ITEM, BLOCK, and STRING. Each of these is described below.

ARRAYS

The favored input form (there are several) is one very similar to NAMELIST. Each input array has a unique name. To make an input to an array, one simply spells out the array name, appends an equal sign, and follows that with the data items to be entered into the array. For example, input for the coarse mesh boundaries for the y direction could look like:

```
YMESH= 0 1 2 4I 3 8,
```

The above input would enter values of 1,2,3,4,5,6,7, and 8 centimeters for the mesh boundaries. Note that a FIDO-like interpolate was used. In general, all the FIDO operators may be used in numeric entry. A complete list of the valid operators is given in Table III.1.

Unlike NAMELIST however, an array name CANNOT use a subscript. The operators A, S, and E described in Table III.1 may be used for this addressing function.

Data items within an array are separated by blanks or commas. In general, blanks may be used freely throughout except within a data item, within an array name, or between an array name and its equal sign.

Numeric DATA ITEMS

Numeric data items follow a FORTRAN input convention. For example, all of the following are valid entries for the number ten:

```
10, 1.0+1, 1E1, 10.0
```

If a decimal point is not entered, it is assumed to be after the right-most digit.

Hollerith DATA ITEMS

Hollerith data items follow a FORTRAN variable name convention in that they are composed of up to six characters, the first of which must be alphabetic with the rest alphanumeric. However, special characters may be included if the data item is surrounded by double quotes. Operators may NOT be used with Hollerith data items.

BLOCKS

Arrays are entered in groups called blocks. A block consists of one or more arrays (in any order) followed by the single character T. Thus T is the block delimiter.

STRINGS

Arrays may be broken into smaller pieces called strings. Strings are delimited with a semicolon(;). The user should be aware of the arrays that require string input. Strings are frequently used to input information by row rather than for the whole 2-d array at once. The code dictates this, the user has no choice.

Comments

A slash (/) may be used to enter comments in the input stream. After a slash is read no further processing of that card is done.

For complete details of the free field input, the user is referred to the ONEDANT manual (Ref. 2).

Table III.1 Valid Input Operators.

nR d	REPEAT the data item d, n times
nI d	INTERPOLATE (linear) n data items between data item d and the next data item.
F d	FILL the rest of the data string with the data item d.
nY m	STRING REPEAT. Repeat the previous m strings, n times.
nL d	INTERPOLATE LOGARITHMICALLY n data items between d and the next d.
nZ	ZERO. Enter the value zero n successive times.
nS	SKIP. Skip the next n data items.
nA	ADDRESS. Set the pointer to the n-th data item in the array.
nQ m	SEQUENCE REPEAT. Enter the last m entries, n more times.
nG m	SEQUENCE REPEAT WITH SIGN CHANGE. Same as the Q option but the sign of the m entries is changed every repeat.
nN m	SEQUENCE REPEAT INVERT. Same as the Q option but the order of the m entries is inverted each repeat.
nM m	SEQUENCE REPEAT INVERT WITH SIGN CHANGE. Same as N option but the sign is also changed every repeat.
nX	COUNT CHECK. Causes code to check the number of entries in the current string so far, against the number n.
E	END. Skips to the end of the string.

Note: The operator character must always be appended directly to n. d or m need not be immediately adjacent to the operator character.

IV. TWODANT INPUT SPECIFICATIONS

A. Input Overview

The full TWODANT input consists of a title card section, followed by six blocks of free field input. The title card section is not free field. Any input referred to as a block uses the free field input form.

Block-I consists of basic control and dimensional information that allows efficient packing of the array data. This information also allows checking of the lengths of arrays supplied by interface files.

Block-II contains the geometric information.

Block-III consists of the nuclear data specifications.

Block-IV contains mixing information.

Block-V contains the rest of the input needed for specifying the flux calculation.

And lastly, Block-VI contains the edit (i.e. report writing) specifications.

A cross-section library may optionally be placed between Blocks III and IV, if it is in card image form. TWODANT supports many library formats and so the library may or may not be in free field format depending upon the option chosen.

A full input would then look like that diagrammed in Figure IV.1 on the following page.

Figure IV.1   General Input Structure

```
Figure IV.1   General Input Structure
-------------------------------------

         **********************
        *                      *
       *   Title Card Count     *
       *                        *
       **************************

         **********************
        *                      *
       *   Title Card(s)        *
       *                        *
       **************************


         **********************
        *                      *
       *   Block-I              *
       * (Controls and Dims)    *
       *                  T     *
       **************************

         **********************
        *                      *
       *   Block-II             *
       *    (Geometry)          *
       *                  T     *
       **************************

         **********************
        *                      *
       *   Block-III            *
       *    (Nuclear Data)      *
       *                  T     *
       **************************

                **********************
               *                      *
              *   Cross sections on    *
              *   cards (optional)     *
              *                        *
              **************************

         **********************
        *                      *
       *   Block-IV             *
       *    (Mixing)            *
       *                  T     *
       **************************

         **********************
        *                      *
       *   Block-V              *
       *    (SOLVER Input)      *
       *                  T     *
       **************************

         **********************
        *                      *
       *   Block-VI             *
       *    (EDIT Input)        *
       *                  T     *
       **************************
```

B. MINI-MANUAL

On the following few pages is given a complete list of the input names, expected array sizes, and order within the array. No description of the array contents is given in this MINI-MANUAL as full details are given in later sections and also in the ONEDANT manual. The MINI-MANUAL is intended to serve as a quick reference for the knowledgeable user.

In both the MINI-MANUAL and in the detailed sections which follow, a shorthand form is used to indicate the size and order of the array that the code expects. This information is enclosed in square brackets immediately after the array name. Essential features are:

1. A single entry in the brackets is the array length.

2. No brackets at all indicates a simple variable (i.e. an array of unit length).

3. A dash (-) in the brackets indicates an arbitrary length.

4. A semicolon (;) indicates that the input for that array is expected in strings. To the left of the semicolon is the string length. To the right of the semicolon is the number of strings in the array.

5. If the number of strings is shown as a product, the order is important. The leftmost quantity must be exhausted first, then, the next one to the right is varied. For example, the array name for the full spatial source distribution is shown as:

   SOURCF [IT;JT*NMQ]

   - where IT is the number of fine meshes in the X-direction, JT is the number of fine meshes in the Y-direction, and NMQ is the number of input source moments. For this array, the first string is composed of the P0 source values for each x mesh point in the first y mesh. The next string is the P0 source values in the second y mesh. This process is repeated in all y meshes. Then starting again with the first y mesh, the P1 source values are given. After all P1 values are given, the P2 values follow. Continue until all NMQ moments are specified.

   Note: Usually, the quantities within brackets will have already been specified in the input. Sometimes, however, a quantity is derived from the array input itself. For instance, in this particular case, NMQ is not an input quantity, rather, the code counts the number of strings and then, knowing IT and JT, deduces what NMQ must have been.

```
  **********************
 * Title Card Control   *
*   (3I6 Format)        *
*   NHEAD,NOTTY,NOLIST  *
*************************

  **********************
 * Title Card(s)        *
* ------------------    *
*     (IF NHEAD>0)      *
*************************

  **********************
 * Block-I:Ctrls & Dims *
* ---------------------  *
*                        *
*  IGEOM                 *
*  NGROUP                *
*  ISN                   *
*  NISO                  *
*  MT                    *
*  NZONE                 *
*  IM                    *
*  IT                    *
*  JM                    *
*  JT                    *
*                        *
*  MAXLCM                *
*  MAXSCM                *
*                        *
*  IDIMEN                *
*                        *
*  --  --  --  --  --    *
*                        *
*  NOFGEN                *
*  NOSOLV                *
*  NOEDIT                *
*                        *
*                        *
*  NOGEOD                *
*  NOMIX                 *
*  NOASG                 *
*  NOMACR                *
*  NOSLNP                *
*  NOEDTT                *
*  NOADJM                *
*                 T      *
*************************
```

```
  **********************
 * Block-II:Geometry    *
* --------------------   *
*                        *
*   XMESH [IM+1]         *
*   YMESH [JM+1]         *
*   XINTS [IM]           *
*   YINTS [JM]           *
*   ZONES [IM;JM]        *
*                        *
*                 T      *
*************************

  **********************
 * Block-III:Cross-Sect *
* --------------------   *
*                        *
*                        *
*   LIB   valid:ODNINP   *
*               XSLIB    *
*               ISOTXS   *
*               GRUPXS   *
*               BXSLIB   *
*               MACRXS   *
*                        *
*   LNG                  *
*                        *
* -- -- -- -- -- -- --   *
*  Rest of this block    *
*  needed only for card  *
*  image libraries.      *
*                        *
*  MAXORD                *
*  IHM                   *
*  IHT                   *
*  IHS                   *
*  IFIDO  (valid:0/1/2)  *
*  ITITL                 *
*  I2LP1                 *
*  SAVBXS                *
*  KWIKRD  (default:1)   *
*                        *
*  NTPI [NISO]           *
*  NAMES [NISO]          *
*  EDNAME [IHT-3]        *
*  VEL [NGROUP]          *
*  EBOUND [NGROUP+1]     *
*  CHIVEC [NGROUP]       *
*                    T   *
*************************

    **********************
   * iff LIB=ODNINP,      *
  * insert BCD card-image *
  * cross sections here   *
  ************************

  **********************
 *  Block-IV: Mixing    *
* -------------------    *
*                        *
*  MATLS  [-;MT]         *
*  ASSIGN [-;NZONE]      *
*                        *
*  PREMIX [-;-]          *
*                        *
*  MATNAM [MT]           *
*  ZONNAM [NZONE]        *
*                 T      *
*************************
```

```
 ***********************
* Block-V: SOLVER       *
* --------------------  *
*                       *
* IEVT                  *
* ISCT                  *
* ITH                   *
* IBL                   *
* IBR                   *
* IBT                   *
* IBB                   *
*                       *
* EPSI                  *
* IITL                  *
* IITM                  *
* OITM                  *
* ITLIM                 *
*                       *
* FLUXP                 *
* XSECTP                *
* FISSRP                *
* SOURCP                *
*                       *
* --- Flux Guess ------ *
*                       *
*  INFLUX               *
*                       *
* --- Quadrature ------ *
*                       *
*  IQUAD                *
*                       *
*  WGT [MM]             *
*  MU  [MM]             *
*  ETA [MM]             *
*                       *
* --- Miscellaneous --- *
*  NORM                 *
*  BHGT                 *
*                       *
*  CHI  [NGROUP;M]      *
*                       *
*  DEN  [IT;JT]         *
*     -or-              *
*  DENX [IT] and/or     *
*   DENY [JT]           *
*                       *
* --------------------  *
*                       *
*---Volumetric Source---*
*                       *
* INSORS                *
*                       *
* SOURCE [NGROUP;NMQ]   *
*     -or-              *
* SOURCX [IT;NMQ] and   *
* SOURCY [JT;NMQ]       *
*     -or-              *
* SOURCX [IT;NMQ] and   *
* SOURCY [JT;NMQ] and   *
* SOURCE [NGROUP;NMQ]   *
*     -or-              *
* SOURCF [IT;           *
*       JT*NGROUP*NMQ]  *
*     -or-              *
* SOURCF [IT;JT*NMQ] and*
* SOURCE [NGROUP;NMQ]   *
*                 T     *
*************************
```

```
 ***********************
* Block-VI: EDIT        *
* --------------------  *
*                       *
* PTED                  *
* ZNED                  *
*                       *
* POINTS [K], K<IT+1    *
* EDZONE [IT;JT]        *
*                       *
* ICOLL [K], K<NGROUP+1 *
* IGRPED                *
*                       *
* POWER                 *
* MEVPER                *
*                       *
* RZFLUX                *
* BYVOLP                *
* AJED                  *
*                       *
* EDXS [K], K<NEDT+1    *
* RESDNT                *
* EDISOS [K], K<NISO+1  *
* EDCONS [K], K<NISO+1  *
* EDMATS [K], K<MT+1    *
* XDF [IT]              *
* YDF [JT]              *
*                       *
* RSFE [NGROUP;-]       *
* RSFX [IT;-]           *
* RSFY [JT;-]           *
* RSFNAM [-]            *
*                       *
* MICSUM [-]            *
* IRSUMS [-]            *
*                       *
*                 T     *
*************************
```

C. Full Input Details

The following pages of this section (i.e. IV.C) give details for each of the input arrays. All valid TWODANT arrays are discussed in this section in detail complete enough to form the input. However, the beginning user, particularly one unfamiliar with discrete-ordinates codes, may find that he is missing some information of a background nature. For that type of information, the user is referred to the ONEDANT manual (Ref. 2).

First, here are a few general instructions:

1. All six of the input blocks are normally included. Block-I is always required but any of the other five blocks may be omitted under the proper conditions. The input module reads each block in turn and from it generates one or more binary interface files. The interface files drive the SOLVER and EDIT modules. Thus, if the user wants no edits, the Block-VI input may be omitted. Then with no interface file, the EDIT module will not be executed. Alternatively, if the interface file is available from another source, the corresponding block of input may be omitted. For instance, Block-II describes the geometry. The input module normally writes this information to the GEODST interface file. If the GEODST file is available from another source or a previous run, the Block-II input may be omitted.

2. The general theme of the ONEDANT input is observed in that arrays that are not needed are not entered. Presence of an array indicates that it should be used. Thus, for example, if the density array is entered (DEN array), the cross section at each mesh point should be modified accordingly. No separate switch need be set to say that the calculation should be done. To eliminate the density modification, simply remove the DEN array from the input.

3. The arrays in general are grouped in the input instructions according to function. Thus, for example, the volumetric source is found on a single page under an appropriate page heading. If one has no volumetric source, one simply skips to the next page of instructions. There is no need to read further on that page.

4. In an adjoint run, none of the groupwise input arrays should be inverted. The code will externally identify all groups by the physical group number, not by the calculational group number (the calculational group number is in inverse order). Thus, the user interface should be consistently in the physical group order.

5. The use of information within square brackets to indicate the size of arrays and the order within those arrays is the same as described in the introduction to the MINI-MANUAL (previous section).

6. With the exception of the SOLVER (Block-V) input, any ONEDANT input is also a valid TWODANT input. Thus, users familiar with ONEDANT input may freely use any options they are currently using in ONEDANT. However, within the SOLVER block input, there are some valid ONEDANT input arrays which are not valid for TWODANT. The code will not reject them but the desired action will not take place. Such arrays will be mentioned here in the details and indicated as unimplemented. These arrays are not shown at all in the MINI-MANUAL. Only the functional arrays are shown there.

7. New users reading these instructions for the first time and unfamiliar with the TWODANT input may find it helpful to follow the sample input in Appendix B while reading this section.

```
******************************
*     TITLE CARD DETAILS     *
******************************
```

Card 1: · Title Card Control (format 3I6)

Word	Name	Comments
1	NHEAD	Number of title (header) cards to follow
2	NOTTY	Suppress output to on-line user terminal? 0/1 = no/yes (default=no)
3	NOLIST	Suppress listing of all card image input? 0/1 = no/yes (default=no)

Cards 2 thru NHEAD+1: Title Cards (format 12A6)

NHEAD title cards containing descriptive comments

{ Title card section always required}

```
                 ******************************
                 *          BLOCK-I           *
                 *  CONTROLS AND DIMENSIONS   *
                 ******************************


 Name           Comments
-------       ---------------------------------------

IGEOM         Geometry: 6/7 = X-Y/R-Z
NGROUP        Number of energy groups
ISN           Sn order to be used
NISO          Number of physical isotopes on the basic input cross-
              section library
MT            No. of physical materials to be created
NZONE         No. of geometric zones in problem
               (each neutronically homogeneous)
IM            Number of coarse mesh intervals in the X (or R) direction
IT            Total number of fine mesh intervals in the X (or R) direction
JM            Number of coarse mesh intervals in the Y (or Z) direction
JT            Total number of fine mesh intervals in the Y (or Z) direction
 {all above input always required}

MAXLCM        Length of LCM desired (default=50000)
 {optional}
MAXSCM        Length of SCM desired (default=20000)
 {optional}

IDIMEN        1/2 = 1d/2d Dimension control for the input and edit modules
  {optional}              (TWODANT default: 2d, but this may be overridden
                           if one wishes to, say, run only the input
                           module to generate files for ONEDANT)

   ---     ---     ---     ---     ---     ---     ---     ---

   Note:  The above input is all that is necessary in Block-I for
          a full run.  The controls below allow partial runs and
          are otherwise not needed.  For full details on their use,
          see chapter VIII of the ONEDANT manual (Ref. 2).

          Default on all these variables is no.

NOFGEN        0/1 = no/yes suppress further input module execution
NOSOLV        0/1 = no/yes suppress solver module execution
NOEDIT        0/1 = no/yes suppress edit module execution

NOGEOD        0/1 = no/yes suppress writing GEODST file even though
                           geometry cards may be present
NOMIX         0/1 = no/yes suppress writing mixing files even though
                           mixing cards may be present
NOASG         0/1 = no/yes suppress writing ASGMAT file even though
                           block-IV may be present
NOMACR        0/1 = no/yes suppress writing MACRXS file even though
                           both block-III and block-IV may be present
NOSLNP        0/1 = no/yes suppress writing SOLINP file even though
                           block-V may be present
NOEDTT        0/1 = no/yes suppress writing EDITIT file even though
                           block-VI may be present
NOADJM        0/1 = no/yes suppress writing ADJMAC file even though
                           an adjoint calculation is called for
```

```
******************************
*          BLOCK-II          *
*      GEOMETRY DETAILS      *
******************************
```

Name	Comments
XMESH [IM+1]	X coordinates of coarse mesh edges.
YMESH [JM+1]	Y coordinates of coarse mesh edges.
XINTS [IM]	Number of fine meshes in each coarse x mesh
YINTS [JM]	Number of fine meshes in each coarse y mesh
ZONES [IM;JM]	Zone number for each coarse mesh. This array defines the geometric zones to which cross-section materials are assigned.

{all above input required if this Block is entered}

Note: The information entered in this block is written to the CCCC standard interface file GEODST.

```
******************************
*         BLOCK-III          *
*    NUCLEAR DATA DETAILS    *
******************************
```

Name	Comments
LIB {always}	Source of the cross-section data. Enter as a data item one of the following six character words.
LNG {optional}	Number of the last neutron group. Used only to separate neutrons from gammas in the edits.

Word	Description
ISOTXS	CCCC standard isotope ordered binary cross-section file.
XSLIB	Card image BCD library supplied in a separate file named XSLIB.
ODNINP	Card image BCD library follows after this block of input (after the T of Block-III).
GRUPXS	CCCC standard group ordered cross-section file.
BXSLIB	Binary form of XSLIB or ODNINP libraries from a previous run.(more efficient)
MACRXS	Use existing files named MACRXS for SOLVER module, SNXEDT for EDIT module. These files were created in a previous run. Under this option any remaining Block-III input and, unless otherwise specified in Block-I, any PREMIX and MATLS input in BLock-IV will be ignored.

Note: The CCCC standard for files ISOTXS and GRUPXS does not allow the inclusion of the 2L+1 term in the higher order scattering cross section. However, if you have a nonstandard file, you may override by setting I2LP1=-1. TWODANT will then accept cross sections containing the 2L+1 term.

```
*******************************
*        BLOCK-III            *
*    NUCLEAR DATA (cont.)     *
*******************************
```

Note: The remaining arrays in block-III are used only if the source of cross-section data is XSLIB or ODNINP.

Name	Comments
MAXORD	Highest Legendre order in the scattering tables.
IHM	Number of rows in a cross-section table.
IHT	Row number of the total cross section.
IHS	Row number of the self-scatter cross section.
IFIDO	Format of the cross-section library: 0/1/2 = Los Alamos(6E12)/fixed-field FIDO/ free-field FIDO
ITITL	0/1 = no/yes A title card precedes each table.
I2LP1	0/1 = no/yes Higher order scattering cross sections on the library contain the 2L+1 term.
SAVBXS	0/1 = no/yes Save the binary form of the card image library XSLIB or ODNINP for use in a subsequent run. Saved on file BXSLIB.
KWIKRD	Process fixed-field FIDO-format, card-image BCD library with fast processor at the sacrifice of error checking? 0/1 = no/yes (default=yes)
NAMES [NISO]	Hollerith name for each of the input isotopes. Can be used later in mixes. (default names are: ISO1, ISO2, . . . etc.)
EDNAME [IHT-3]	Hollerith name for each of the EDIT cross-section positions used in the cross-section edits. These are the positions before the absorption cross section in the cross-section table. (default names are : EDIT1, EDIT2, . . .etc.)
NTPI [NISO]	Number of legendre scattering orders for each isotope in the library. (default=MAXORD+1)
VEL [NGROUP]	Speeds for each group. Needed only for alpha calcs.
EBOUND [NGROUP+1]	Energy group boundaries. Presently unused.
CHIVEC [NGROUP]	Chi vector (fission fraction born into each group). Used for every isotope. Can be overridden by zone dependent CHI input in Block-V.

--- --- --- --- --- --- --- ---

Card image BCD libraries may be entered in one of the three forms indicated in the IFIDO input. All three forms share the following features: Cross sections are entered in a table optionally preceded by a title card. A table consists of IHM*NGROUP entries, so different Legendre orders are in different tables. Order within group G is as follows:

. . . ABS,NU-SIGF,TOTAL,. . .,GtoG, G-1toG, G-2toG, etc.

In the Los Alamos format, the table is entered with a standard FORTRAN 6E12 format.

In the fixed field FIDO format, entries are made in six twelve-column fields. Each twelve-column field is divided into three subfields, a two-column numeric field, a one-column character field, and a nine-column numeric field. See a DOT or ANISN manual for details if you are not familiar with this input. The last field in a table must have the character T in the character position. No array identifier should be used.

In the free field FIDO form, entries do not have to be in designated columns. Rather, the rules specified in the previous FREE FIELD INPUT BRIEFING apply. Each table in this form is also terminated with the character T. No array identifier (i.e. array name with appended equals sign) should be used.

```
******************************
*          BLOCK-IV          *
*       MIXING DETAILS       *
******************************
```

Input to this block is unchanged from that shown in the ONEDANT manual. However, a short summary of the two main mixing arrays is included here for quick reference. Normally, these two arrays are required and, in most problems, would be the only arrays in this block.

The key entities used in specifying the cross-section spatial distribution are coarse mesh, zone, isotope, and material.

The basic geometry of the problem is defined with the coarse meshes specified in Block-II. The geometric areas called zones are also defined there using the ZONES array; The ZONES array designates which coarse meshes are contained in each zone.

Here in Block-IV, we mix cross sections and assign them to the zones created above. The cross sections found on the input library belong, by definition, to isotopes, no matter what their true nature. These isotopes may then be mixed to form materials, using the MATLS= array. Materials are then assigned to zones using the ASSIGN= array.

--- --- --- --- --- --- --- ---

The general form of a MATLS mix instruction is shown below:

MATLS= mat1 comp1 den1, comp2 den2, ...etc.... ;

where - mat1 is the hollerith name of the first material and comp1, comp2, and so on are the hollerith names of its components which have densities of, respectively, den1, den2, and so on. Additional materials may be defined in subsequent strings. Each string may contain as many components as necessary(actual limit=500). A component is usually an isotope from the library, but may also be a temporary material created by the PREMIX= array. The form of the PREMIX= array is identical to that of the MATLS= array. The difference in treatment is that the temporary materials created by PREMIX= exist only long enough to complete the mixing; they are not available for assignment to geometric zones, nor are they available for use in material edits.

Short form: MATLS= ISOS

This form specifies that isotope number 1 is to be used for material number 1, isotope number 2 is to be used for material number 2, and so on.

--- --- --- --- --- --- --- ---

The general form of the ASSIGN instruction is shown below:

ASSIGN= zone1 mat1 vol1, mat2 vol2, ...etc.... ;

where - zone1 is the hollerith name to be used for the first zone (the one specified with numeral 1 in the ZONES array). mat1, mat2, and so on are the hollerith names of the materials that will be present in this zone with, respectively, the volume fractions vol1, vol2, and so on.

Short form: ASSIGN= MATLS

This form specifies that material number 1 is to be assigned to zone number 1, material number 2 to zone number 2, and so on.

--- --- --- --- --- --- --- ---

Note: The information entered in the MATLS= array is written to the CCCC standard files NDXSRF and ZNATDN. Information entered in the ASSIGN= array is written to the code dependent file ASGMAT.

Additional Mixing Arrays

On the previous page, isotopes, materials, and zones were identified by their hollerith names. Optionally, they may be referred to by their ordinal number. Thus, 2 for an isotope name would call for the second isotope on the library.

THE HOLLERITH NAME FORM IS HIGHLY RECOMMENDED. It provides the most straight forward, most understandable form. If the hollerith name form is used, the input arrays below are unneeded.

Using the hollerith name form in one array and the numeric name form in another array is particularly discouraged. However, should one wish to use the numeric form in the MATLS= and/or ASSIGN= arrays, and then subsequently associate hollerith names with the ordinal numbers, one can use the following arrays to do so. This situation could arise, if for some reason, one wanted to use material numbers in the MATLS= array, but use hollerith material names in the ASSIGN= array.

MATNAM [MT] {optional}	Hollerith material names for Materials. Used only if the mat1 name used in the MATLS= array was integer. First entry in MATNAM array is the hollerith name for Material number 1, second entry is the hollerith name for Material number 2, etc.
ZONNAM [NZONE] {optional}	Hollerith zone names for Zones. Used only if the zone name entry in the ASSIGN= array was integer. First entry in the ZONNAM= array is the hollerith name for Zone number 1, second entry is the hollerith name for Zone number 2, etc.

Concentration Searches

Concentration searches are not presently implemented in TWODANT so the mixing arrays pertaining to the search mentioned in the ONEDANT manual should be ignored.

BLOCK-V
SOLVER DETAILS

Name	Comments	
IEVT	Calculation type: 0/1 = homogeneous source/k-eff	
ISCT	Legendre order of scattering	
ITH	0/1 = direct/adjoint calculation	
IBL	Left bdry condition: 0/1/3 = vacuum/reflective/white	
IBR	Right bdry condition: 0/1/3 = vacuum/reflective/white	
IBT	Top bdry condition: 0/1/2/3 = vacuum/reflective/periodic/white	
IBB	Bottom bdry condition: 0/1/2/3 = vacuum/reflective/periodic/white	
EPSI	Convergence precision (default=0.001)	
EPSO	This variable unused by TWODANT	
IITL	Maximum no. of inner iterations per group at first (default chosen by code)	
IITM	Maximum number of inners allowed when near convergence (default chosen by code)	
OITM	Maximum no. of outer iterations (default=20)	
ITLIM	Number of seconds time limit (default=unlimited)	
FLUXP	0/1/2 = no/isotropic/ all moments	Final flux print
XSECTP	0/1/2 = no/mixed/all	Cross-section print
FISSRP	0/1 = no/yes	Fission rate print
SOURCP	0/1/2/3 = no/unnormalized/ normalized/both	Source print
GEOMP	Not used by TWODANT	
RAFLUX	Not yet implemented in TWODANT	
ANGP	Not yet implemented in TWODANT	

--- --- --- --- --- --- --- ---

Note: TWODANT presently has no search capability, so the remaining input on this page should be ignored.

IPVT	0/1/2 = none/k-eff/alpha	Type of eigenvalue to search for in a concentration search
PV	Value of k-eff or alpha to search to	
EVM	Amount to change concentration by in first step of search	
XLAL	Lambda lower limit for search	
XLAH	Lambda upper limit for search	
XLAX	Lambda convergence criterion for second and subsequent search steps	
POD	Parameter oscillation damper (default=1.)	

```
*******************************
*    SOLVER DETAILS(cont.)    *
*          Flux start         *
*******************************
```

```
 Name                 Comments
-------         ---------------------------------------

 INFLUX         0/1 = no/yes Read flux start from the RTFLUX file.

  ----- Note: There presently is no card input flux guess available.
```

```
******************************
*    SOLVER DETAILS(cont.)    *
*    Quadrature and Misc.     *
******************************
```

***** Quadrature Details *****
{optional}

Name	Comments
IQUAD {optional}	Source of quadrature constants -3/1 = SNCONS file/Built-in constants (default=1)
WGT [MM]	Quadrature weights. . Presence of these
MU [MM]	Mu cosines. . arrays overrides
ETA [MM] {together optional}	Eta cosines. . the IQUAD input

***** Miscellaneous *****
{all optional}

Name	Comments
NORM	Normalize the fission source rate to this value when IEVT.GE.1 or normalize the inhomogeneous source rate to this value when IEVT.LT.1. NORM=0 means no normalization. (Integral of source rate over all angle, space, and energy = NORM).
BHGT	Buckling height to use to correct for leakage. Units are centimeters.
CHI [NGROUP;M]	Fission fraction born into each group. Enter by zone up to M zones. Succeeding zones (i.e. zones M+1 through NZONE) will use the CHI values from zone M.
DEN [IT;JT]	Density to use at each fine mesh point.
-or-	
DENX [IT] and/or	Density to use at each fine x-mesh (default=1).
DENY [JT]	Density to use at each fine y-mesh (default=1).
	Note: In this latter form, the density factor, DEN(i,j), at mesh interval (i,j) is computed as follows: DEN(i,j) = DENX(i)*DENY(j)

```
******************************
*   SOLVER DETAILS(cont.)    *
*    Volumetric Source       *
******************************
```

Name	Comments
INSORS	0/1 = no/yes Read source from interface file FIXSRC.

----- For card input source, choose one of the following options:

Option 1:

SOURCE [NGROUP;NMQ]	Source spectrum for each of NMQ moments. (Spatial distribution is assumed to be flat with value unity)

Option 2: (input both arrays)

SOURCX [IT;NMQ]	X (or R) spatial distribution for each moment.
SOURCY [JT;NMQ]	Y (or Z) spatial distribution for each moment. (Spectrum is assumed to be flat with value unity)

Option 3: (input all three arrays)

SOURCE [NGROUP,NMQ]	Source spectrum.
SOURCX [IT;NMQ]	X (or R) spatial distribution for each moment.
SOURCY [JT;NMQ]	Y (or Z) spatial distribution for each moment.

Option 4:

SOURCF [IT;JT*NGROUP*NMQ]	Spatial distribution for each row, group, and moment.

Option 5: (input both arrays)

SOURCE [NGROUP;NMQ]	Source spectrum.
SOURCF [IT;JT*NMQ]	Spatial distribution for each row and moment.

Note: Only in option 4 is the complete pointwise source array, SOURCF(i,j,g,m), given. In all other cases, it must be formed from the lower dimension arrays that are input. That calculation is done by forming the product of those arrays. Thus, in option 3, where the source spectrum, SOURCE(g,m), and the spatial distributions SOURCX(i,m) and SOURCY(j,m) are given (for moment m), the full source at mesh point (i,j) in group g for moment m is calculated as follows:

```
SOURCF(i,j,g,m) = SOURCE(g,m)*SOURCX(i,m)*SOURCY(j,m)
```

```
*******************************
*    SOLVER DETAILS(cont.)    *
* Special Boundary Conditions *
*       {UNIMPLEMENTED}       *
*******************************
```

Boundary Sources

ONEDANT supports isotropic and angle dependent boundary sources. These boundary conditions have not yet been implemented in TWODANT and the following arrays which are ultimately intended for that purpose will be ignored in TWODANT:

```
SILEFT   SIRITE          SALEFT   SARITE
SITOP    SIBOTT          SATOP    SABOTT
```

Albedoes

ONEDANT supports albedo boundary conditions on all boundaries. These boundary conditions have not yet been implemented in TWODANT and the following arrays which are ultimately intended for that purpose will be ignored in TWODANT:

```
LBEDO    RBEDO           TPBEDO   BTBEDO
```

```
*****************************
*        BLOCK VI           *
*   EDIT INPUT DETAILS      *
*****************************
```

***** Spatial Specifications *****

Name	Comments
PTED {always}	0/1 = no/yes Do edits by fine mesh
ZNED {always}	0/1 = no/yes Do edits by zone (i.e. edit zone, not SOLVER zone. See EDZONE input below.)
POINTS [<IT] {optional}	Fine mesh point (or interval) numbers at which point edits are desired. Must be in ascending order. USED ONLY IF PTED=1. (Default= all points)
EDZONE [IT;JT] {optional}	Edit zone number for each fine mesh interval. USED ONLY IF ZNED=1. (default= SOLVER coarse mesh interval numbers, see XMESH array, Block-II)

***** Energy Specifications *****

Name	Comments
ICOLL [NBG] {optional}	Edit energy group collapsing option. Number of SOLVER energy groups in each EDIT broad group. The NBG entries must sum to NGROUP. (Default = 1 energy group per EDIT broad group)
IGRPED {optional}	Print option on energy groups:- 0/1/2/3 = Print energy group totals only/ Print broad groups only/ Print broad groups only(same as 1)/ Print both broad groups and totals (Default = 0)

***** Power Normalization *****

Name	Comments
POWER {optional}	Normalize to POWER megawatts. All printed reaction rates and the fluxes on files RTFLUX and RZFLUX (if requested) will be normalized.
MEVPER {optional}	MeV released per fission (default=210 MeV). This value will be used along with the calculated fission rate to determine the power. For the power calculation, TWODANT needs to know which cross section is the fission cross section. It uses the one from the library that has the name N-FISS. If one uses an ISOTXS or GRUPXS library that designation will automatically be made (see Table IV.1). But if one uses a card-image library, either ODNINP or XSLIB, then the name N-FISS must be entered in the proper place in the EDNAMES array.

```
******************************
*          BLOCK VI          *
*  EDIT INPUT DETAILS(cont.) *
******************************
```

***** Cross-Section Edits *****

Name	Comments
EDXS [<NEDT]	Cross-section types to be used in forming reaction rates. May be entered by integer (denoting edit position of desired cross-section type) or by the hollerith name of the cross-section type. See Table IV.1 for the available names. NEDT is the total number of Edit cross-section types available from the input cross-section library. (default = all shown in Table IV.1) Note: The cross-section types specified in this array apply to any or all of the following edit forms: RESDNT, EDISOS, EDCONS, EDMATS.
RESDNT {optional}	0/1 = no/yes Do edits using the resident macroscopic cross section at each point. See note below.
EDISOS {optional}	Hollerith names of the isotopes to be used in forming Isotopic reaction rates. The ordinal number may alternatively be used but is not recommended. (default = none)
ECONS [<NISO+1] {optional}	Hollerith names of the isotopes to be used in forming resident Constituent (partial macroscopic) reaction rates. The ordinal number may alternately be used but is not recommended. (default = none) See note below.
EDMATS [<MT+1] {optional}	Hollerith names of materials to be used in forming Material (macroscopic) reaction rates. The ordinal number may alternately be used, but is not recommended. See note below. (default = none)
XDF [IT] YDF [JT] {optional}	Fine mesh density factors for the X(or R) and Y(or Z) directions, respectively. The density factor is used to multiply resident Constituent (see EDCONS), Material macroscopic (see EDMATS), and Resident Macroscopic (see RESDNT) reaction rates only. See note below. (default = all values unity)

Note: If density factors were used in SOLVER to modify the cross sections at each mesh, the same density factors must be provided in the XDF and/or YDF arrays here in this block as well. The density factor at mesh point (I,J) is computed as:

XDF(I)*YDF(J)

***** Miscellaneous *****

Name	Comments
RZFLUX {optional}	0/1 = no/yes Write the CCCC standard zone flux file RZFLUX
BYVOLP {optional}	0/1 = no/yes Printed point reaction rates will have been multiplied by the mesh volume.
AJED {optional}	0/1 = no/yes Regular (forward) edit/Adjoint edit Regular edit uses the RTFLUX scalar flux file; adjoint edit uses the ATFLUX flux file.

```
******************************
*  EDIT INPUT DETAILS(cont.) *
*     (Block VI)             *
******************************
```

***** Response Function Edits *****

Name	Comments
RSFE [NGROUP;M] {required if user input response functions are desired}	Response function energy distribution for each of the M different response functions desired. The number of different response functions is arbitrary (but must be fewer than 500). Data are entered as M strings,.each with NGROUP entries beginning with group 1.
RSFX [IT;M] RSFY [JT;M] {optional}	Response function X(or R) distribution for M functions. Response function Y(or Z) distribution for M functions. Data are entered as M strings of IT or JT entries beginning with mesh point 1. (default=1.0) Note: M-th response function at space point (I,J) and energy group G is computed as RSFX(I,M)*RSFY(J,M)*RSFE(G,M)
RSFNAM [M] {optional}	Hollerith names for the user-input response functions specified above. M is arbitrary but must be less than 500. (default = RSFP1,RSFP2,...RSFPM)

***** Reaction Rate Summing *****

Name	Comments
MICSUM [<500 sums] {optional}	Cross-section reaction rate summing specifications. The MICSUM array is a packed array with data entered as follows: A set of Isotope numbers or names is given, followed by a set of cross-section type position numbers or names (see Table IV.1). These sets are delimited with an entry of 0 (zero). Reaction rates are calculated for each Isotope specifed for each cross-section type specified and summed to form the first sum. The next two sets of data are used to form the second sum, etd. Up to 500 sums can be specified. (see Section VII.D.1 in the ONEDANT manual)
IRSUMS [<500 sums] {optional}	Response function reaction rate summing specifications. The IRSUMS array is input as follows: A set of response function numbers or names is entered and the set delimited with an entry of z (zero). Reaction rates are calculated using these response functions, and the rates are summed to form the first sum. The next set of data is used to form the second sum, etc. Up to 500 sums can be specified. (see Section VII.D.2. in the ONEDANT manual)

TABLE IV.1

EDIT CROSS-SECTION TYPES BY POSITION AND NAME

CROSS-SECTION INPUT VIA ISOTXS/GRUPXS			CROSS-SECTION INPUT VIA BCD CARD-IMAGES		
Type	EDIT Position	Name (a)	Type	EDIT Position	Name (a)
chi	1	CHI...	not used	1	CHI...
nu-fission	2	NUSIGF	nu-fission	2	NUSIGF
total	3	TOTAL.	total	3	TOTAL.
absorption	4	ABS...	absorption	4	ABS...
(n,p)	5	N-PROT	1 (b)	5	EDIT1. (c)
(n,d)	6	N-DEUT	2 (b)	6	EDIT2. (c)
(n,t)	7	N-TRIT	3 (b)	7	EDIT3. (c)
(n,alpha)	8	N-ALPH	.	.	.
(n,2n)	9	N-2N..	.	.	.
(n,gamma)	10	N-GAMM	.	.	.
fission	11	N-FISS	N=IHT-3	4+N	EDITN. (c)
transport	12	TRNSPT			

Notes:

a. Names are six character hollerith. A period within a name denotes a blank.

b. Denotes position (row) in the cross-section table. All cross sections in rows (positions) 1 through IHT-3 in the cross-section library are EDIT cross sections chosen by the user.

c. These are the default names that may be overridden with the user-option names in the EDNAMES array of Block III.

APPENDIX A

FILE DESCRIPTIONS

The user is referred to the ONEDANT manual (Ref. 2) for a description of the code-dependent interface files. Those files are identical to the ones used here in TWODANT.

APPENDIX B

SAMPLE INPUT

On the following page is a small but complete sample problem input. It is a two group calculation of the eigenvalue of an R-Z model of a sodium cooled fast reactor. The geometric model contains two zones, a cylindrical core zone surrounded by a reflector zone. PO cross sections for each isotope are entered in the input stream after Block-III. These isotopes are subsequently mixed to form the materials STEEL, FUEL, AND SODIUM. These materials are then assigned with appropriate volume fractions to the CORE and REFLECTOR zones.

In the edit input, the code is asked to give reaction rate totals for each edit zone. The reaction rates desired are the default ones, that is, CHI, NUSIGF, TOTAL, ABS, and EDIT1 (EDIT1 is the default name for the first position in the card-image library).

Note the use of comments (using the slash, /) to organize and describe the input.

```
    2     0     0
SAMPLE PROBLEM FOR TWODANT USER'S GUIDE
STANDARD K CALCULATION,  ALL INPUT BY MEANS OF CARD-IMAGES
/   GEOMETRY       - R,Z
/   CROSS SECTIONS - 2 GROUP, ISOTROPIC SCATTER
/                    ISOTOPE DATA ON CARDS,  LOS ALAMOS (DTF) FORMAT
/   MIXING         - ISOTOPES MIXED TO MAKE MATERIALS NAMED STEEL,
/                    FUEL, AND SODIUM
/                  - MATERIALS ASSIGNED TO MAKE ZONES NAMED CORE
/                    AND REFLECTOR
/   SOLVER         - CARD INPUT SUPPLIED
/   EDITS          - ZONE EDITS FOR RESIDENT MATERIALS
/
/ -------------------- BLOCK I --------------------
    IGEOM=7, NGROUP=2, ISN=4 NISO=7  MT=3 NZONE=2  IM=2 IT=25
    JM=3  JT=30  IDIMEN=2     T
/
/ -------------------- BLOCK II (GEOMETRY) --------------------
    XMESH=0.0,30,45     XINTS=15,10   YMESH=0,15,60,75   YINTS=5,20,5
    ZONES= 2R2;  1,2;  2R2      T
/
/ -------------------- BLOCK III (CROSS SECTIONS) --------------------
    LIB= ODNINP
    MAXORD=0  IHM=6  IHT=4  IHS=5  IFIDO=0  ITITL=1
    NAMES= "O-16"  "NA-23"  FE  CR  NI  "PU-239"  "U-238"
/
/ ****** SINCE LIB=ODNINP, THE CROSS SECTION LIBRARY IN CARD-IMAGES
/  WILL BEGIN IMMEDIATELY FOLLOWING THE BLOCK III TERMINAL "T".
/   NOTE THAT A TITLE CARD PRECEDES EACH CROSS-SECTION
/   BLOCK (SINCE ITITL=1). ******
/
  T
OXYGEN-16  (O-16)    SAMPLE 2 GROUP LMFBR CROSS SECTIONS
      0.000       0.010       0.000       2.000       1.600       0.000 O16/1
      0.000       0.000       0.000       3.600       3.600       0.390 O16/2
SODIUM     (NA-23)   SAMPLE 2 GROUP LMFBR CROSS SECTIONS
      0.000       0.002       0.000       1.900       1.500       0.000 NA23/1
      0.000       0.005       0.000       4.000       3.995       0.398 NA23/2
IRON       (FE)      SAMPLE 2 GROUP LMFBR CROSS SECTIONS
      0.000       0.008       0.000       2.100       1.700       0.000 FE/1
      0.000       0.010       0.000       4.500       4.490       0.392 FE/2
CHROMIUM   (CR)      SAMPLE 2 GROUP LMFBR CROSS SECTIONS
      0.000       0.013       0.000       2.450       2.150       0.000 CR/1
      0.000       0.020       0.000       5.000       4.980       0.287 CR/2
NICKEL     (NI)      SAMPLE 2 GROUP LMFBR CROSS SECTIONS
      0.000       0.080       0.000       2.400       2.000       0.000 NI/1
      0.000       0.030       0.000       8.000       7.970       0.320 NI/2
PLUTONIUM  (PU-239)  SAMPLE 2 GROUP LMFBR CROSS SECTIONS
      1.900       1.950       6.270       4.800       2.000       0.000 PU239/1
      1.600       2.500       4.800      12.000       9.500       0.850 PU239/2
URANIUM    (U-238)   SAMPLE 2 GROUP LMFBR CROSS SECTIONS
      0.300       0.400       0.900       4.700       3.000       0.000 U238/1
      0.000       0.500       0.000      13.000      12.500       1.300 U238/2
/
/ ******* END OF CROSS-SECTION DATA ******
/ **** NOTE THAT THERE IS NO TERMINAL "T" SINCE THE CROSS SECTIONS ARE
/      IN LOS ALAMOS (DTF) FORMAT (IFIDO=0) ****
/
/ -------------------- BLOCK IV (MIXING) --------------------
    MATLS= STEEL, FE .05, CR .016,  NI 0.01;
           FUEL  "PU-239" .0103,  "U-238" .0103   "O-16" .0412;
           SODIUM  "NA-23" .025
    ASSIGN= CORE    FUEL  .35, SODIUM .4, STEEL .25;
            REFLEC  SODIUM .7, STEEL  .3              T
/
/ -------------------- BLOCK V (SOLVER) --------------------
    IEVT=1   ISCT=0  IBR=0  IBT=0  IBB=0
    NORM=1     FLUXP=1  XSECTP=2  FISSRP=1
    CHI=0.6,0.4; 0.7, 0.3     T
/
/ -------------------- BLOCK VI (EDITS) --------------------
    ZNED=1, RESDNT=1, T / *** ZONE EDIT FOR RESIDENT MATERIALS
```

APPENDIX C

CCF ACCESS AND EXECUTION

The TWODANT code is presently maintained at the Central Computing Facility (CCF) at the Los Alamos National Laboratory. The following paragraphs show how to access and execute the code at that facility under the CTSS systems. No LTSS version is presently being maintained.

Access

To access the code, do a GET under MASS from the directory:

```
MASS GET /CT1GREEN/TWODANT/TDNddmmm
```

The name of the executable controllee is of the form:

```
TDNddmmm
```

where dd is the day of the month, and mmm is a three letter abbreviation of the month (e.g. TDN30JUN). For debug purposes, the symbol table is included as a part of the controllee.

This manual will be found in the file named TDNMANUAL, also under the same directory. It may be sent directly to a printer (first column should be used as a carriage control), or it may be accessed by any of the text editors.

Execution

Execution of the code is obtained by entering the controllee file name (e.g. TDN30JUN); the input file will be assumed to be on the file named ODNINP and the output will be on the file named ODNOUT. Alternatively, the execute line may be of the form:

```
TDN30JUN  I=infile O=outfile / t p
```

where infile is the input file name and outfile is the output file name.

APPENDIX D

MFE ACCESS AND EXECUTION

The TWODANT code is presently maintained at the National Magnetic Fusion Energy Computer Center at Livermore, California. The following paragraphs show how to access and execute the code at that facility on the CTSS system. No LTSS version is presently being maintained.

Access

To access the code, do a READ with FILEM per below:

```
FILEM / t v

READ .TWODANTC TDNddmmm   (on a CRAY only)
```

The name of the executable controllee is of the form:

```
TDNddmmm
```

where dd is the day of the month, and mmm is a three letter abbreviation of the month (e.g. TDN30JUN). For debug purposes, the symbol table is included as a part of the controllee.

This manual will be found in the file named MANUAL, also under the same directory. It may be sent directly to a printer (first column should be used as a carriage control), or it may be accessed by any of the text editors.

Execution

Execution of the code is obtained by entering the controllee file name (e.g. TDN30JUN); the input file will be assumed to be on the file named ODNINP and the output will be on the file named ODNOUT. Alternatively, the execute line may be of the form:

```
TDN30JUN  I=infile O=outfile / t p
```

where infile is the input file name and outfile is the output file name.

REFERENCES

1. R. D. O'Dell, "Standard Interface Files and Procedures for Reactor Physics Codes, Version IV," Los Alamos Scientific Laboratory report LA-6941-MS (September 1977).

2. R. D. O'Dell, F. W. Brinkley, and D. R. Marr, "User's Manual for ONEDANT: A Code Package for One-Dimensional, Diffusion-Accelerated, Neutral-Particle Transport," Los Alamos National Laboratory report LA-9184-M (February 1982).

3. R. E. Alcouffe, "Diffusion Synthetic Acceleration Methods for the Diamond-Difference Discrete-Ordinates Equations," Nucl. Sci. Eng. 64, 344 (1977).

4. R. E. Alcouffe, "The Multigrid Method for Solving the Two-Dimensional Multigroup Diffusion Equation," Proc. Am. Nucl. Soc. Top. Meeting on Advances in Reactor Computations, Salt Lake City, Utah, March 28-31, 1983, Vol. 1, pp 340-351.

☆U.S. GOVERNMENT PRINTING OFFICE: 1984-0-776-026/4040

Printed in the United States of America
Available from
National Technical Information Service
US Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

Microfiche (A01)

Page Range	NTIS Price Code	Page Range	NTIS Price Code	Page Range	NTIS Price Code	Page Range	NTIS Price Code
001-025	A02	151-175	A08	301-325	A14	451-475	A20
026-050	A03	176-200	A09	326-350	A15	476-500	A21
051-075	A04	201-225	A10	351-375	A16	501-525	A22
076-100	A05	226-250	A11	376-400	A17	526-550	A23
101-125	A06	251-275	A12	401-425	A18	551-575	A24
126-150	A07	276-300	A13	426-450	A19	576-600	A25
						601-up*	A99

*Contact NTIS for a price quote.