ANSTO/E734

ANSTO/E734

ansto

AUS98 - THE 1998 VERSION OF THE AUS MODULAR NEUTRONICS CODE SYSTEM

edited by

G. S. ROBINSON and B. V. HARRINGTON

July 1998

ISBN 0 642 59971 8

ISSN 1030-7745

REPORT E

AUSTRALIAN NUCLEAR SCIENCE
AND TECHNOLOGY ORGANISATION

LUCAS HEIGHTS RESEARCH LABORATORIES

AUS98 - THE 1998 VERSION OF THE AUS MODULAR NEUTRONICS CODE SYSTEM

edited by

G. S. ROBINSON and B. V. HARRINGTON

ABSTRACT

AUS is a neutronics code system which may be used for calculations of a wide range of fission reactors, fusion blankets and other neutron applications. The present version, AUS98, has a nuclear cross section library based on ENDF/B-VI and includes modules which provide for reactor lattice calculations, one-dimensional transport calculations, multi-dimensional diffusion calculations, cell and whole reactor burnup calculations, and flexible editing of results. Calculations of multi-region resonance shielding, coupled neutron and photon transport, energy deposition, fission product inventory and neutron diffusion are combined within the one code system.

This report gives details of all system aspects of AUS and all modules except the POW3D multi-dimensional diffusion module.

ISBN 0 642 59971 8
ISSN 1030-7745

The following descriptors have been selected from INIS Thesaurus to describe the subject matter of this report for information retrieval purposes. For further details please refer to IAEA-INIS-12 (INIS: Manual for Indexing) and IAEA-INIS-13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

A CODES; BURNUP; COMPUTER PROGRAM DOCUMENTATION; COLLISIONS; COMPUTER GRAPHICS; CRITICALITY; CROSS SECTIONS; ENERGY DEPOSITION; FAST REACTORS; FISSION; FISSION PRODUCTS; FORTRAN; GROUP CONSTANTS; ITERATIVE METHODS; KERMA; MULTIGROUP THEORY; NEUTRON DIFFUSION EQUATION; NEUTRON FLUX; NEUTRON TRANSPORT; NEUTRON SPECTRA; NUCLEAR DATA COLLECTIONS; ONE-DIMENSIONAL CALCULATIONS; PHOTON TRANSPORT; REACTOR KINETICS; REACTOR LATTICES; REACTOR PHYSICS; RESONANCE INTEGRALS; PERTURBATION THEORY; THERMAL REACTORS; THREE-DIMENSIONAL CALCULATIONS; TWO-DIMENSIONAL CALCULATIONS

EDITORIAL NOTE

The Australian Nuclear Science and Technology Organisation (ANSTO) replaced the Australian Atomic Energy Commission (AAEC) on 27 April 1987. Reports issued after April 1987 have the prefix ANSTO with no change of the symbol (E, M, S or C) or the numbering sequence.

PREFACE

This document gives a complete description of the system aspects of AUS and all significant modules except the POW3D multi-dimensional neutron diffusion module which has been recently published. In describing the 1998 version of AUS, this document effectively replaces many previously published code descriptions. The organisation of the document follows that of the previous publications in that an updated version of each of those publications is included here as an independent component of the report. The first component, "A Guide to the AUS System", includes an overall description which still refers to previous publications of individual major modules. This was preferred to giving no references at all. The "Guide" also includes descriptions of minor modules and interactive plotting modules as appendices. The major modules are included independently in the order one might use them in undertaking a complete calculation. The introductory remarks about the AUS system have been removed from these components.

The major changes from the previous AUS publications are the inclusion of a cross-section library based on ENDF/B-VI, the addition of the MICBURN module for controlling whole reactor burnup calculations, and changes to the system as a consequence of moving from IBM mainframe computers to UNIX workstations.

The AUS system has been developed at Lucas Heights over the past 25 years and is the result of the endeavours of many people. Significant contributions at various times have been made by Dr J. M. Barry, Dr B. E. Clancy, Dr G. Doherty, Mrs B. V. Harrington, Dr J. P. Pollard and Mr G. S. Robinson.

Table of Contents

A GUIDE TO THE AUS MODULAR NEUTRONICS CODE SYSTEM

. . . . . aus
1. INTRODUCTION 1
2. REPORT CONVENTIONS 1
3. BASIC CONCEPTS OF THE SYSTEM 2
4. CURRENT MODULES 3
5. AUSYS FOR LINKING MODULES AND LOADING DATA 6
6. THE AUS COMMAND 12
7. CONVENTIONS FOR THE USE OF DD NAMES 13
8. STANDARD LINKAGES 14
9. STANDARD PATHS 14
10. CODING AN AUS MODULE 15
11. REFERENCES 18
APPENDIX A - AUS CROSS-SECTION DATA POOLS (XSLIB) 20
APPENDIX B - AUS GEOMETRY DATA POOLS (GEOM) 30
APPENDIX C - AUS FLUX DATA POOLS (FLUXA AND FLUXB) 32
APPENDIX D - AUS STATUS DATA POOLS 35
APPENDIX E - A SAMPLE MODULE 40
APPENDIX F - MINOR MODULES OF THE AUS SYSTEM 44
APPENDIX G - INTERACTIVE PLOTTING MODULES OF THE AUS SYSTEM . . . 51

MIRANDA – A MODULE BASED ON MULTIREGION RESONANCE THEORY FOR GENERATING CROSS SECTIONS

. . . . . miranda
1. INTRODUCTION 1
2. OVERVIEW OF THE CODE AND LIBRARY 2
3. USE OF AUS DATA POOLS 4
4. MODIFICATIONS TO THE RESONANCE TREATMENT 4
5. INPUT DATA 5
6. REFERENCES 20
Table 1. Neutron Group Boundaries for AUS *endfb6* Library 22
Table 2. Photon Group Boundaries 24
Table 3. Nuclides in the AUS *endfb6* Library 24
Table 4. Fission Products in the AUS *endfb6* Library 26

ANAUSN – ONE-DIMENSIONAL MULTIGROUP SN TRANSPORT THEORY MODULE

anausn
1. INTRODUCTION 1
2. ZONING AND PROBLEM SPECIFICATION 1
3. REPORT CONVENTIONS 2
4. INPUT 2
5. OUTPUT 9
6. CODE STRUCTURE 10
7. REBALANCE PROCEDURES 10
8. INVOCATION OF ANAUSN 12
9. REFERENCES 12

ICPP — COLLISION PROBABILITY MODULE

icpp
1. INTRODUCTION 1
2. BRIEF DESCRIPTION OF COLLISION PROBABILITY ROUTINES 1
3. INPUT DESCRIPTION 3
4. REFERENCES 6

EDITAR – REACTION RATE EDITING AND CROSS-SECTION AVERAGING MODULE

editar
1. INTRODUCTION 1
2. USE OF AUS DATA POOLS 1
3. CALCULATION OF LEAKAGE FROM A CELL 2
4. FORMULAE USED IN AVERAGING 2
5. INPUT DESCRIPTION 2
6. REFERENCES 10

CHAR — NUCLIDE BURNUP MODULE

char
1. INTRODUCTION 1
2. DESCRIPTION 1
3. INPUT SPECIFICATION 3
4. REFERENCES 8

MICBURN – MODULE TO CONTROL REACTOR BURNUP USING MICROSCOPIC NEUTRON CROSS SECTIONS

. . . . . micburn
1. INTRODUCTION 1
2. GENERAL DESCRIPTION 1
3. INPUT SPECIFICATION 3
4. REFERENCES 12

BURNMAC – MODULE TO PERFORM REACTOR BURNUP USING MACROSCOPIC NEUTRON CROSS SECTIONS

. . . . . burnmac
1. INTRODUCTION 1
2. GENERAL DESCRIPTION 1
3. INPUT SPECIFICATION 2
4. REFERENCES 6

AUSED – MODULE FOR EDITING AUS CROSS SECTION DATA POOLS

. . . . . aused
1. INTRODUCTION 1
2. CONVENTIONS 1
3. FREE STYLE INPUT DATA 2
4. INPUT LAYOUT 3
5. PRELUDE DATA 3
6. AN AUSED RUN 4
7. AUS CROSS SECTION FILE DATA 5
8. SUBGROUP PARAMETER FIT 9
9. EDITING OF OUTPUT 13
10. EXAMPLES 15
11. ACKNOWLEDGEMENTS 17
12. REFERENCES 18

A GUIDE TO THE AUS MODULAR NEUTRONICS CODE SYSTEM

1. INTRODUCTION

The AUS system of neutronics computer codes was developed initially for fission reactor core calculations, and has been applied to a wide range of fast and thermal reactor types. The original system [Robinson 1975] has undergone considerable development. The most significant of these was the extension to cover fusion blanket calculations [Robinson 1984] which required the inclusion of photon production, photon interaction and nuclear heating. The revised system has sufficient detail on nuclear data to be used for many other neutron and photon applications as well as fission reactor cores and fusion blankets.

AUS is a modular system in which the computer codes (modules) may be executed in a very flexible manner. The modules communicate through well-defined data sets on disc. The AUS modules are quite large, which leads to some duplication of function but also provides an easier system for users. Although user input has not been entirely standardised, most problem specifications have to be entered once only. The present system has a cross-section library derived from ENDF/B-VI but previous libraries derived mainly from ENDF/B-IV are also available. AUS currently includes modules which provide for lattice calculations, one-dimensional transport calculations, one-, two- and three-dimensional diffusion calculations, and burnup calculations. Provision is made for the flexible editing of results and interactive graphics.

The accuracy of AUS using ENDF/B-VI and ENDF/B-IV libraries for fission reactor calculations has been established by comparison with a wide range of fast and thermal reactor benchmark experiments [Robinson 1993]. Detailed comparison of the AUS resonance calculation with more detailed collision probability and Monte Carlo calculations using the point cross section files from which the AUS libraries were generated has shown that the accuracy in resonance captures is usually about 1% [Robinson 1977, 1985b]. This may deteriorate when there are large resonance overlap effects between resonances in different nuclides.

The AUS system was written originally for IBM mainframe computers and had some system components which were written in Assembler language. The current version is written in Fortran 77 except for some minor C routines. It has been run on a number of computers with UNIX operating systems and has also been run on a PC with DOS. AUS is currently being run mainly on a Silicon Graphics Power Challenge. This report summarises the available modules and provides details of all system aspects of AUS. This includes the supervisor program, that part of user input which controls the calculation sequence, and the interface data sets. Sufficient detail is given to enable other modules to be added to the system. Details of some of the minor modules of the system are given in Appendix F and interactive graphics modules are outlined in Appendix G. This report effectively replaces both earlier guides [Robinson 1975, 1987].

2. REPORT CONVENTIONS

The following conventions have been used in this report:

- module names are given in UPPER CASE,
- types of data file are given in UPPER CASE,
- Fortran is given in UPPER CASE,
- information to be reproduced exactly is given in **bold**,
- items that the user will replace with an actual value are given in *italics*,
- computer file names are given in *italics*,
- examples and defaults are given in `constant width` type,
- omission of some data is indicated by ...,
- optional items are given inside [ ],

- items from which the user will choose are listed in a column inside { },
- a set of input lines is represented by a *descriptive phrase in italics*,
- ***bold italics*** are used for emphasis.

3. BASIC CONCEPTS OF THE SYSTEM

A ***data pool*** is a set of data with a defined structure which is used to pass information between modules. The structure and content of the data set must also be well documented. The following data pools are currently defined:

- XSLIB for cross sections,
- GEOM for geometry description,
- FLUXA and FLUXB for fluxes, and
- STATUS for isotopic compositions and spatial smearing factors.

A ***module*** is any computer program which is self-contained except for a user-supplied input stream and input/output of data via data pools. Any program could be considered an AUS module, but it is the interaction of the program with other modules of the scheme which gives meaning to the term. A module does need to indicate errors by setting a condition code. The file containing an object module is executed using a Fortran call to the SYSTEM subroutine with the file name as a shell command. The execution of a module in this way is referred to within AUS as ***linking*** the module.

A ***path*** (or ***step***) controls the sequence in which modules are linked to perform the required calculation. It also controls the data sets to be used by the modules. A path is written in a mixture of Fortran and MACRO statements which are translated into Fortran using a preprocessor. The path program is then compiled and executed under the supervision of the AUSYS program. As it is written in Fortran, the path itself can perform subsidiary calculations. The sequence in which the modules are linked, and the data sets on which they operate, can thus be programmed to depend on any calculated result made available to the path.

The ***aus shell procedure*** executes in turn a program to preprocess the path, the Fortran 77 compiler and link editor to prepare an executable for the path, and finally the path program. The shell procedure also includes references to AUS subroutine libraries, the default cross section library, and data sets specific to individual modules. These libraries, data sets and executables are kept in a standard AUS directory which users should set to the environment variable $AUS.

Modules open files by referring to them indirectly using ***DD names*** which are mostly of the three- or four-character form *ddn*. An AUS user may specify that particular named files are to be used for any of these DD names, otherwise a temporary file will be used. It is a function of the path to make the association of the DD names *ddn* (and hence the required files) with Fortran unit numbers, for each module that is linked. The supervisor program AUSYS assists in making this association by retrieving the standard set of unit numbers and DD names for each module from the file *linklib* in the standard AUS directory. This directory is referred to in this report as simply *aus*, so that the above file becomes *aus/linklib*.

The ***input stream*** for AUS consists of input to AUSYS and to each module to be linked by the path. The input stream is broken up into blocks by records of the form ***dd**n*, beginning in column one. The set of records following each ***dd**n* record is loaded onto the data set with the DD name *ddn*. Each of these data sets *ddn* consists of the user input data to a particular module. The only restriction on this user input to a module is that it cannot contain records with ***dd** in columns 1 to 3. System data must always be loaded on the *dd1* file and consist of

- control information for AUSYS;
- a source of a non-standard path or a request to retrieve a standard path from *aus/pathlib*; and

optionally

- any data required as direct input to the path program.

The ***AUSYS program*** supervises and supports the execution of a path. AUSYS consists of a set of subroutines which are loaded together with the path to form an executable which controls the AUS calculation. Requests from the path to link a module are interpreted and the module is executed using the SYSTEM function of Fortran 77. AUSYS contains a number of utility routines which may be called directly by the path. The AUS and ANSTO subroutine libraries (Section 5.3.7) are also available. AUSYS is also used to maintain the system data sets *aus/pathlib* and *aus/linklib*.

The sample input stream given below illustrates some of the features of the scheme. The six records following `*dd1` form the path program which, in turn, links the three modules MIRANDA, ANAUSN and EDITAR with input following `*dd2`, `*dd3` and `*dd4`, respectively. The three modules together form a lattice calculation. The six records following `*dd1` could be replaced by the single record `step mae`, as they constitute a standard path:

```
aus << 'eof'
*dd1
step *
      link miranda
      link anausn(1,3)
      link editar(1,4)
      end
stop
*dd2
head trx1 ref endf-202
defn fuel u235 6.253-4 u238 4.7205-2
defn void o 1.-5
defn clad al .06025
defn mod h2o .03338
reqd fuel void clad mod u235 u238
rm 0 5*.0983 .0127 .0711 5*.0745836 0
reg 1(1)5 fuel 6 void 7 clad 8(1)12 mod
resreg 6 .4915 .0838 .372918 -1 smear 5*1 2 2 5*3
buck b3 5.7-3
output p3
groups 26 -1 .5(0.5)2.5(1.5)10. 12.5 14.3 15.3 16.2(0.4)20.2 21. 23.
start
stop
*dd3
trx
aus nl 1 nsn 6 end
*dd4
buck b3 5.7-3  search off
groups 2 1 15 26  output rr  start
```

4. CURRENT MODULES

4.1. Introduction

This section describes briefly the available modules, comments on their standard usage and outlines some features of the user input which are common to all modules. Further information on the accessing of data pools by each module is given in Section 8.

4.2. MIRANDA — Cross-section Generation

The MIRANDA module [Robinson 1986c, 1977] includes a multiregion (for slabs, cylinders or clusters) resonance calculation of the subgroup type, and a cell average B_n flux solution for preliminary group condensation. The cross-section library is an AUS data pool with temperature dependence and subgroup parameters fitted to tabulated resonance integrals as a function of potential scattering. The current library *aus/endfb6* which has 200 neutron groups and 37 photon groups was generated from ENDF/B-VI (Robinson 1993). It corresponds to revision 1 of the ENDF/B-VI files. Two older libraries which both have data obtained mostly from ENDF/B-IV are also available. The original library, *aus/endfb*, had 128 neutron groups only. The more recent library, *aus/endf200g* has the same group structures as *aus/endfb6*.

In lattice calculations, data from MIRANDA for each region of a cell are passed to one of the transport theory modules to continue the calculation. A large number of groups should be retained for large cells as the MIRANDA flux solution is homogeneous.

4.3. ANAUSN — 1D, SN Transport

The ANAUSN module [Clancy 1982] is a general purpose, one-dimensional discrete ordinate module with anisotropic scattering. Fixed source, reactivity and criticality search calculations can be performed with this module which is the standard module for cylindrical cells and all 1D global transport calculations.

4.4. ICPP — Isotropic Collision Probability

The ICPP module [Robinson 1985a] includes both approximate and accurate numerical methods for calculating collision probabilities in slab, cylindrical, spherical and cluster geometries. The module was developed particularly for many-group, few-region calculations of group condensation spectra. For many-region cell calculations, ICPP is preferred for slabs and clusters, and ANAUSN for other geometries.

4.5. POW3D — Multidimension Diffusion Including Kinetics

The POW3D module [Harrington, Pollard & Barry 1996] is a general purpose, three-dimensional diffusion code which includes feedback-free kinetics. The module includes general criticality search options and extensive editing facilities, including perturbation calculations and graphics. POW3D uses a finite difference method in which fluxes are calculated at mesh points rather than mesh intervals.

4.6. AUSIDD — 1D Diffusion

The AUSIDD module (Appendix F) is a simple one-dimensional diffusion module intended for global calculations of condensation spectra, which is complementary to the multidimensional POW3D module. It differs from the POW3D module in using a finite difference scheme with mesh points at the centre of the mesh intervals.

4.7. EDITAR — 1D and 2D Editing

The EDITAR module [Robinson 1986a] provides reaction rate editing and cross-section condensation facilities following a transport calculation. The module includes a B_n calculation of cell leakage, provision for bilinear weighting and perturbation calculations, and access to the STATUS data pool for nuclide concentrations and spatial smearing factors.

4.8. CHAR — Lattice and Global Burnup

The CHAR module [Robinson 1986b] uses an analytic method to solve the nuclide burnup equations and the cross-section library provides the chain mechanisms. Lattice cell or global calculations may be undertaken in as many regions as desired. The normal function of the module is simply to update the STATUS data pool of nuclide compositions following a burnup step. However, macroscopic cross sections may be formed using data on the cell flux distribution from the status data pool. This feature is used to perform global microscopic burnup calculations.

4.9. MICBURN — Global Burnup and Fuel Management

The MICBURN module [Robinson 1994] is used in conjunction with CHAR and a module such as POW3D to perform global burnup calculations in which the nuclide composition in each reactor zone is calculated using reaction rates from the global flux calculation. A zone is normally a fuel element or portion thereof. The functions of MICBURN are to initialize the calculation, control the calculation sequence, change fuel, and provide summary printed output as the calculation proceeds. The microscopic cross sections on which the computation is based are generated in a previous cell burnup calculation and may be irradiation dependent.

4.10. BURNMAC — Global Burnup

The BURNMAC module [Robinson 1986b] provides a simple alternative method for global burnup calculations and assumes that macroscopic cross sections are a function of irradiation only. Cross sections as a function of irradiation are obtained from previous lattice burnup calculations using the CHAR module.

4.11. AUSED — Cross-section Editing

The AUSED module [Harrington 1976] loads or edits an AUS cross-section data pool. It may be used to maintain the basic libraries but is mostly used for temporary cross-section modification on the user's cross-section data pools. The conversion from tabulated resonance integrals to subgroup parameters is also carried out by this module.

4.12. PLOTFL2 — Plotting

The PLOTFL2 module [Clancy 1978] plots fluxes and reaction rates based on a one-dimensional flux data pool. The user may input a simple formula to be used in forming the reaction rate. PLOTFL2 may be used for plotting when running background jobs but its interactive function has been replaced by AUSPLOT (Appendix G).

4.13. ORNL — Form Cross Sections for Standard Transport Codes

The ORNL module (Appendix F) produces cross sections from an AUS cross-section data pool either as ASCII in ANISN-style input format or as a binary AMPX file. This provides a rather weak link between the AUS system and widely available transport codes such as ANISN [Engle 1967], DORT [Rhodes & Child 1988] and KENO [Petrie & Landers 1984].

4.14. Modules For Data Pool Manipulation

Three modules, MERGEL, JOINER and EXPAND (Appendix F), are available for the manipulation of data pools. The most useful of these is MERGEL which combines two AUS cross-section data pools. The other two modules are more specialised, their use being confined to some burnup applications.

4.15. Interactive Plotting Modules

Two interactive modules used for plotting are invoked directly rather than being linked within AUS. The module PLOTXS (Appendix G) is used for plotting cross sections from any AUS XSLIB including the main libraries. The module AUSPLOT (Appendix G) is used for plotting fluxes, and reaction rates following a 1D, 2D or 3D flux calculation.

4.16. Input Features Common To Most Modules

Though user-supplied input data to the modules of the AUS scheme have not been standardised, most modules have a similar style of input. The modules use the input routine SCAN [Bennett & Pollard 1967] to read free format data which are grouped into entries. Each entry consists of a keyword to indicate the data type followed by a string of data. The same entries are used in a number of modules. Default values are available for many input parameters so that data requirements are kept to a minimum for standard calculations.

Some of the standard features used with the SCAN routine are as follows:

- Data are entered in columns 1 to 72 only.
- Keywords may be up to eight alphanumeric characters.
- Floating point data may be given in abbreviated form; for example `1,1.,1.E+0,1.-0,1.E 0` are equivalent.
- Repeat notation is, for example, `4*0. = 0. 0. 0. 0.`
- Increment notation is, for example, `1(2)7 = 1 3 5 7.`
- Special characters may be used in most contexts at the discretion of the user to improve readability.
- A record with an * in column 1 is a comment record.

5. AUSYS FOR LINKING MODULES AND LOADING DATA

5.1. Introduction

The functions of the AUSYS supervisor are split between a preprocessor program, PREPARE, and a set of routines forming the main part of AUSYS which are loaded with the path program to form an executable which controls the execution of modules. The functions of the preprocessor are:

- process the input stream by copying the data following each ***dd**n* record onto the data set with DD name *ddn*;
- print the complete input stream, but this may be stopped at any point by including **noprint** on an ***dd** record (and turned on again by including **print**);
- analyse the path program in sufficient depth to recognise the AUS MACRO statements and replace them by subroutine calls;
- provide facilities for maintaining the system files *aus/pathlib* and *aus/linklib*.

The path is retrieved from *aus/pathlib* or entered as part of the system input following the ***dd1** record. After preprocessing, the path is compiled and loaded with the AUSYS routines which provide a suitable environment for its execution. When a path requests that a module be linked, AUSYS calls the Fortran SYSTEM function to execute the module. After the module is executed, the path resumes execution as if the module were merely a subroutine. To the path, the only difference between calling a subroutine and linking a module is that all data sets opened using the **alter** MACRO statement (Section 5.3.3) are closed when the module is linked.

5.2. Input Layout

Input to AUSYS is always included following a ***dd1** record in the AUS input stream. The input is in the form of directives in free format and input in standard Fortran layout for non-standard

paths. In the following description, data in **bold** to be provided by the user must be entered from column 1. For normal AUS calculations there are three different forms:

A.

step *named*

where *named* is the name of a standard path to be obtained from *aus/pathlib*.

B.

step *

Set of path statements

stop

where the set of path statements is a complete non-standard path.

C.

seek *named*

step *

Set of path statements

stop

where *named* is the name of a set of standard subroutines to be retrieved from *aus/pathlib* which, when combined with the input records, forms a path.

The **step** records may contain an additional integer m. If it is included, the first $m-1$ links to modules are ignored. This option assists in restarts of calculations provided that the required data pools have been saved.

Any of the three forms may be followed optionally by directives to initialise data sets and also by data which are direct input to the path program. The directive to initialise a data set is

$dd*n* **disp=new**

where **dd***n* is the DD name of a data set which is to be initialised with an end of file mark. This initialisation should be necessary only for existing STATUS data pools which are to be restarted. Further information on the editing features provided by AUSYS for STATUS data pools is given in Appendix D.

Data required as direct input to the path program are given following the directive

$dd99

which must be the last **$dd** directive entered.

5.3. Coding an AUS Path

5.3.1. Summary of the path language

The language is that of the Fortran compiler invoked by the AUS procedure but MACRO statements may be included anywhere in the program. A MACRO statement consists of a keyword followed by a string of characters. The MACRO statement is compiled as a subroutine call with the character string as an argument. For example, the statement

```
link pow3d (1,3)
```

is a MACRO statement resulting in the execution of the POW3D module. The combination of such statements with normal Fortran statements provides flexibility and ease of path coding.

5.3.2. The link MACRO

The **link** MACRO sets up I/O unit assignments for a module and causes the module to be executed. AUSYS returns control to the next statement in the path when the module concludes. The path writer may regard this as a normal subroutine return except that any data sets opened

using the **alter** MACRO will have been closed. AUSYS tests the condition code returned by the module, and terminates the entire calculation if the code is not in the range 1 to 7 inclusive. This range was chosen to avoid having to find and modify all error exits of existing programs used as modules. These limits may be changed, however, by the path (Section 5.3.7).

The form of the MACRO is

link *named* $(m_1, n_1), (m_2, n_2), \ldots$

where *named* is the module name, which is also the name of the file in the AUS directory which contains the required module;

m_i is a Fortran logical unit used in the *named* module and has the form of an integer *ii* (or the form **ft*ii*f001** for compatibility with previous usage);

n_i is a DD name of the type *ddn* which is to be associated with the Fortran unit m_i. The form of n_i may be any of
(a) a positive integer *k* representing the DD name *ddk*,
(b) **dd***k* explicitly,
(c) an **alias** for *ddk* (Section 5.3.5), or
(d) a negative integer -k which means integer j in the k^{th} word in COMMON (Section 5.3.7) giving the DD name *ddj*.

The last indirect form of n_i allows dynamic specification of the data sets which are to be used by a module; for example,

```
      common dummy(20),iunit
      do 1 iunit= 6,10
    1 link pow3d(1,-21)
```

results in execution of the POW3D module using the DD names `dd6` to `dd10` as input in turn.

As all Fortran logical units used by a module should be associated with a DD name (and hence a file), the direct specification of these in the **link** statement becomes tedious. To make path coding easier, the standard unit assignments for each module are stored on *aus/linklib* (Section 8) and are recalled automatically. Only those assignments which are to be changed need to be specified in the **link** statement. There must be a one to one correspondence between Fortran unit numbers and DD names. Two Fortran units cannot be assigned to the same DD name. The normal use that is made of the DD names *ddn* is described in Section 7.

Example

```
link miranda(1,2),(2,dd12),(ft07f001,-27)
```

5.3.3. The alter MACRO

The **alter** MACRO is used to assign Fortran units to DD names for the path itself. The path program may also issue Fortran OPEN statements directly for files not used by a module. The **alter** statement has exactly the same form as the **link** statement except that *named* is not given. The standard assignment (of interest to the normal user) for a path if no **alter** is given is (1, `dd1`). This assignment is maintained by **alter** unless specifically overwritten. As the execution of a **link** MACRO does not affect the unit assignments for the path, an **alter** MACRO may be included at the beginning of a path and this unit assignment used throughout. AUSYS closes these units before each module is linked and opens them again on return from the module. However, other OPEN files are not closed by AUSYS. Thus the path input stream is not closed and reading may continue.

Example

```
alter(2,dd12)
```

5.3.4. The testm MACRO

Because all modules are normally operating system load modules given as members of the directory *aus* , provision of a method for testing changes to modules is required in AUSYS. The **testm** MACRO has the form

testm *mod1=tmod1 mod2=tmod2 ...*

where the first of each pair of arguments gives the name of a current module (maximum of 8 alphanumeric characters) and the second of each pair gives the name of a test module in *aus* , or the full path name of a test module, which is to be linked whenever the current module is requested. An environment variable may also be used. For example :

```
testm  anausn anausnew
```

or

```
testm miranda=/home/gsr/srce/miranda/test
```

or

```
testm miranda=$S/miranda/test
```

5.3.5. ALIAS for DD names

To use a particular data set for any of the AUS data pools, a symbolic parameter may be given the value of a file name using the **aus** command (Section 6). Each of these three-character symbolic parameters is permanently associated with a DD name. As it is much easier for a path writer to think in terms of these symbolic parameters than the DD names of form *ddn*, the symbolic parameter is made an alias for the DD name. The alias can then be used in all **link** and **alter** statements. The set of aliases built into AUSYS is

dd31	*dd33*	*dd34*	*dd40*	*dd35*	*dd36*	*dd37*	*dd41*	*dd38*	*dd39*
lib	*xs1*	*xs2*	*xs3*	*gm1*	*fl1*	*fl2*	*fl3*	*st1*	*st2*

For coding convenience, a path may include or change aliases by using the **alias** MACRO. The alias must be of three characters only.

Example

```
alias (xs4,dd41),(xs1,dd27)
link pow(6,gm1),(8,lib),(18,xs4),(10,xs1)
```

5.3.6. Multiple case facility

AUSYS provides a simple multiple case facility. To use this facility

- the MACRO statement **modify** is included at the beginning of the path;
- a Fortran loop, which includes the **link** MACRO statements to be repeated is coded;
- the characters **mod** (or simply **m**) are included anywhere in columns 73 to 80 on those input records to be replaced; and
- the replacement input records are given in the input block *dd10*.

Once the modify option is on, the system checks the file used as the input stream to each module after it is linked and replaces any records marked as above by the next record in the *dd10* input block. The option may be turned off by using the statement **modify off**. To implement this option, the default unit assignment for AUSYS and the path program has been modified to include

(27, dd*n*), (28, dd10), (29, dd14) ,

where **dd14** is a temporary data set and **dd*n*** is the input block of the last module linked. Words 11 and 12 of blank COMMON are used by this option.

5.3.7. Use of blank COMMON

A path and the AUSYS program share the same blank COMMON area in which the first 20 words are reserved for AUSYS use but are available to the path. A path may extend COMMON, as in the example,

```
COMMON A(1000)
COMMON B(980)
```

The variable A comes after the 20 reserved words.

Each routine in a path has seven variables pre-defined by a built-in COMMON of

COMMON STIME, ICOND, PROG(2), JERR, JFR, LCOND(2), ISKP

where	STIME	is the time available for the AUS step in minutes;
	ICOND	is the condition code returned by the last module executed;
	PROG(2)	is the name (2A4) of the last module executed;
	JERR	is not currently used;
	JFR	is used in association with the SCAN free input routines;
	LCOND(2)	is the permissible range of condition codes (1 and 7 are standard); and
	ISKP	is one more than the number of further **link** statements which are to be ignored.

5.3.8. Routines available to a path

All the standard Fortran functions and subroutines may be used by the path. In addition, routines from ANSTO and AUS subroutine libraries may also be used. The AUS library includes the I/O routines for an AUS cross-section data pool (Appendix A) and the subroutines detailed below.

CALL CLOCK (T)

which returns the time in minutes from the start of the AUS calculation.

CALL EXITA

which returns control to the operating system. AUSYS inserts a call to EXITA before the Fortran END statement in the main routine of the path. This routine enables AUSYS to terminate cleanly and should be used in the path program in preference to EXIT.

CALL PSTAT (IU, IOUT)

which lists on unit IOUT the STATUS data pool that is on Fortran unit IU. Unless IOUT = 6, the output is in a form suitable for reading by AUSYS from the input stream.

CALL PMATS (IU,RAD)

which produces on Fortran unit 2 material definitions suitable for reading by MIRANDA from the input stream. These definitions are extracted from the STATUS data pool on Fortran unit IU for the irradiation value RAD for those materials which have undergone burnup. RAD has the same units as the CHAR module, which are normally watt-days. Linear interpolation in irradiation is used.

CALL LIST (IU, MNXST, MNSP, MNSCAT, MNSCSP, MNP)

which lists the AUS cross-section data pool that is on Fortran unit IU. The remaining arguments are for the ARDT subroutine (Appendix A) and are all equal to 2 for a complete listing.

CALL LISTEN (IU, MNXST, MNSP, MNSCAT, MNSCSP, MNP, MSHORT, MGAP,
MGA, IFNUC, IFN, IFGA)

which is called if a partial list of an AUS cross-section data pool is required. The additional arguments take the following values:

- MSHORT is zero if only the beginning of each record, *i.e.* one printed line, is to be printed,

otherwise the full record is printed.
• MGAP is the number of terms n in a P_{n-1} expansion of photon production which are to be printed and zero gives no photon production.
• MGA is the number of terms in the expansion of photon interaction as for MGAP.
• IFNUC is a vector such that if IFNUC(I)=0, the I^{th} nuclide is not printed.
• IFN is a vector of length equal to the number of neutron groups which takes the following values for each group:
0 no print,
1 print everything,
2 print cross sections only,
3 print scattering matrices only, and
4 print photon production only.
• IFGA is a vector of length equal to the number of photon groups which serves the same function for photons as IFN does for neutrons.

CALL PREMAC (IU, IOPT, JOPT, WTN, WTP, NMAT)

which extracts information relating irradiation and ^{235}U number densities from an ST1 status file for use on the FUEL entry in the BURNMAC module. The arguments are:
• IU is the Fortran unit number of the ST1 file,
• IOPT is 1 if data are required for each fuel material , 0 if the materials are to be mixed;
• JOPT is 1 if the burnup units required are per cent ^{235}U burnup, else 0;
• WTN is a dimensioned variable giving multipliers for the number densities in each fuel material;
• WTP is a dimensioned variable giving multipliers for irradiations originally in $Wdcm^{-3}$ (usually fuel meat volumes and these are used for zero values of WTP) for each fuel material;
• NMAT is the number of fuel materials.

All arguments for PREMAC are supplied by the user, not returned by the subroutine. The normal usage is to use PREMAC to print the required data from a cell burnup calculation and edit the printout for inclusion in the BURNMAC input file. If the defaults are used (WTN and WTP set to 1.), the subroutine returns the ^{235}U number density at each burnup step and the fuel irradiation in $Wdcm^{-3}$ at each burnup step. The multipliers may be used to turn these into the required quantities such as ^{235}U loading in grams per element and *Wd*. WTN may be used to change from number densities in $10^{24}cm^{-3}$ to grams for example. If fuel meat volumes are used (WTP zero), then irradiations will be in Wd.

5.4. Maintenance of the System Files

The system files *aus/linklib* and *aus/pathlib* are sequential unformatted Fortran files which are maintained using the AUSYS program. The file *aus/linklib* contains the standard unit assignments used in linking each module. The file *aus/pathlib* contains the standard paths of the scheme.

Input records which update, load or print the files are included in the system input immediately following the ***dd1** record. The directives **update**, **load**, **stop** and **print** control the available functions. The directive **print** causes the contents of both path and link libraries to be listed. The input required to update or add the *named* path is

update
step *named*
A set of records containing the path program
stop

A set of standard subroutines (to be retrieved using the **seek** directive) is handled in the same way.

To update or add a **link**, the input required is

update
link *named k* $(m_1, n_1), \ldots (m_k, n_k)$
where the form and meaning are the same as for the **link** MACRO except that the number of unit specifications k must be given, an alias may not be used, and Fortran layout (*i.e.* column 6 continuation) may not be used.

The **load** directive is similar to **update** except that a completely new library is created. Any number of **step** directives (each delimited by a **stop**) and **link** directives may be included between the **load** directive and a final **stop** record.

Functions to delete entries or reorganise the data sets have not been included in AUSYS. A file containing the data to be permanently included in the libraries is maintained and used to reload the libraries from time to time.

6. THE AUS COMMAND

Because AUS is normally run under UNIX this description is given in terms of UNIX operation. The AUS shell procedure is written in Bourne shell script and is obtained on the *pion* Silicon Graphics Power Challenge computer simply by using the command **aus**. This may be achieved by including the standard AUS directory, *aus*, in the user's UNIX command path. Note that the AUS system under UNIX is completely in lower case. That means all data as well as file names. Modules are files in the directory *aus* and the file name is identical to the module name. The **aus** procedure includes a reference to each of the AUS system files and to each special file required by a module.

The input to the run is read from the standard input file (*stdin*). Temporary files for the run are created in one of the directories */tmp* or */var/tmp* with names of the form *#.pid.xxxx* where *pid* is the process number of the **aus** command. This allows the user to run several jobs simultaneously. For simple AUS jobs requiring no permanent files, no arguments are required. Thus to run a job **test** , the usage might be

```
aus <<'eof'
Set of input
eof
```

where the above data are stored in an executable file, say **test** , and the job is run using

```
test >& test.out &
```

Note that the system standard output file has been used for printer output in order to have error messages included in the approximately correct position. The last **eof** in a file may be omitted.

The first optional command line argument, *arg1*, is a character string which may be given to identify this run. The remaining optional arguments are used to identify permanent files. A little care is required because AUS modules by default attempt to read from various data pools and, in the case of the STATUS data pool, add to the existing file. The arguments have three different forms. The first is *xxx*= where *xxx* is either an AUS data pool alias such as **xs1** or takes the form **dd***nn* where *nn* is a one or two digit integer. It may also take the form **exe** which refers to the executable formed from the path program and AUSYS routines. This argument specifies that the data pool is not to be temporary and has the name *arg1.xxx* in the current directory. Additionally, if *arg1* starts with "/", the files are in the directory specified by the base component of the

pathname *arg1*. The second form directly specifies the file, using *xxx=dsname* where *dsname* is any file name. If the data pool alias given is **lib** , the file is assumed to be in the main AUS directory rather than the current directory. An environment variable, set to a directory name say, may be used as the first part of appropriate arguments.

The third form specifies whether the file specified by the preceding argument already exists. The arguments are either **disp=new** or **disp=old** if the file must not or must already exist. If neither argument is given, the status is taken to be **unknown** when the file is opened using the Fortran OPEN statement.

EXAMPLE

```
aus  trx1  lib=endf200g  xs3=  disp=new  <<'eof'
```

where the cross section library is to be `endf200g` rather than `endfb6` and **xs3** is to be written on a file *trx1.xs3* which must not already exist.

The default value of the region size for the AUS scheme is 4 megabytes which is adequate for most calculations. The modules of the scheme generally make use of whatever memory is available but may require more than 4 megabytes for some large calculations. The region size may be altered by specifying the symbolic parameter REGION. To increase the region to 8 megabytes, for example, use

```
REGION=8 aus << 'eof'
```

The type of plot and the file on which plots are produced by non-interactive modules may be specified using the environment variables PLOT and PLOTFILE respectively. For example, to produce plots suitable for a laser printer on the file *plotfl.ps*,

```
PLOT=laser PLOTFILE=plotfl.ps aus << 'eof'
```

7. CONVENTIONS FOR THE USE OF DD NAMES

The use made of the data set associated with the name *ddn* is mainly governed by a set of conventions.

The data sets on *dd1* to *dd11* and *dd13* are temporary formatted data sets which are used for user input to modules. The data in the main input stream following an ***ddn** record are automatically loaded onto the *ddn* data set at the start of a calculation. The DD name *dd12* is used by modules for standard formatted output which is card image *i.e.* up to 80 characters. The data sets on *dd14* to *dd16* are further temporary card image data sets which may be used as scratch files by a module or path. The data sets on *dd17* to *dd30* are a corresponding set of unformatted data sets. At present, *dd14* and *dd21* to *dd24* are the only scratch data sets used by modules, except for POW3D which uses *dd15* and *dd17* to *dd30*.

The data sets on *dd31* and *dd33* to *dd41* are used as the data pools of the AUS scheme. The DD name *dd31* is used for a standard AUS cross-section library, and *dd33*, *dd34* and *dd40* are used for cross-section data pools. The DD name *dd35* is used for a geometry data pool, and *dd36* and *dd37* are used for the flux data pools FLUXA and FLUXB, respectively. The DD names *dd38* and *dd39* are used for a pair of STATUS data pools; *dd39* should not normally be saved as it is merely a pointer data set for *dd38*. The DD name *dd41* is used for the flux dump for the POW3D module.

The data sets on *dd53* and *dd56* are special data sets which are each used by one module only.

Further temporary unformatted files may be used with the DD names *dd42 to dd50*. Temporary formatted files may be used with the DD names from *dd57* onwards. It should be noted that files with DD names in the range from *dd17* to *dd50* are opened as unformatted and the remainder as formatted. Normal usage should be consistent with this.

aus: 14

8. STANDARD LINKAGES

The standard association of Fortran logical unit numbers with DD names for each AUS module is stored on *aus/linklib*. Only the required changes then need be given in the **link** MACRO of a path. The standard assignments for each module are tabulated below:

ANAUSN
01 dd2 02 dd12 08 dd21 09 xs1 10 gm1 11 fl2 12 dd22 22 xs2
23 xs3
AUSED
01 dd2 02 dd12 08 lib 10 xs1 11 xs2 04 dd56 07 dd21 12 xs3
AUSIDD
01 dd2 07 gm1 09 fl2 10 xs1
BURNMAC
01 dd2 04 gm1 25 fl1 07 dd3 08 dd14 09 st1 10 xs1 11 xs2
CHAR
01 dd2 04 gm1 25 fl1 26 fl2 07 xs1 08 xs2 09 st1 10 st2
11 dd21 12 dd22 13 dd23 14 xs3
EDITAR
01 dd2 02 dd12 04 st1 25 st2 07 gm1 09 fl2 10 xs1 11 xs2
12 xs3
EXPAND
01 dd2 04 st1 25 st2 10 xs1 11 xs2 26 dd21
ICPP
01 dd2 11 fl2 13 gm1 14 xs1 16 xs2
JOINER
10 xs1 11 xs2 12 xs3
MERGEL
01 dd2 04 st2 10 xs1 11 xs2 29 dd21 12 xs3
MIRANDA
01 dd2 04 st1 25 st2 26 dd21 07 gm1 08 lib 09 fl2 10 xs1
11 xs2 12 xs3 13 dd22 14 dd23 15 dd24
ORNL
01 dd2 02 dd12 08 dd21 09 dd22 10 xs1 11 xs2
PLOTFL2
01 dd2 09 xs1 10 gm1 11 fl2
POW3D
01 dd2 02 dd12 04 dd53 26 gm1 07 dd15 08 lib 09 dd21 10 xs1
11 xs2 14 dd28 18 xs3 19 dd29 20 dd30 21 dd23 22 dd24 23 dd25
24 fl1 25 dd22 61 fl3 62 dd17 63 dd18 64 dd19 65 dd20 66 dd26
67 dd27

9. STANDARD PATHS

Standard path programs may be retrieved from the system data set *aus/pathlib* by entering the following directive in the AUSYS input, *i.e.* after the ***dd1** record:

step *named*

where *named* is the name of a path.

Because the modules of AUS are so large, and paths are normally so simple to write, this feature has proved to be relatively unimportant. The only paths currently available which are of general interest are the simple module sequences detailed below and the MICBURN path [Robinson

1994] used with the MICBURN module. A user's own standard paths may be added to the library and this has been of value.

Single module paths are available for the modules MIRANDA, ANAUSN, ICPP, and POW3D. For example

```
step pow3d
```

is equivalent to

```
step *
      link pow3d
      end
stop
```

There are four paths for cell calculations; `mie` and `mae` are for one-stage calculations, and `mieae` and `mieie` are for two-stage calculations. A two-stage calculation is many-group, few-region followed by few-group, many-region. The paths are

```
step mie    implies:
step *
      link miranda(1,2)
      link icpp(1,3)
      link editar(1,4)
      end
stop

step mieai    implies:
step *
      link miranda(1,2)
      link icpp(1,3)
      link editar(1,4)
      link anausn(1,5),(9,xs3)
      link editar(1,6),
     1 (10,xs3),(12,xs1)
      end
stop
```

```
step mae    implies:
step *
      link miranda(1,2)
      link anausn(1,3)
      link editar(1,4)
      end
stop

step mieie    implies:
step *
      link miranda(1,2)
      link icpp(1,3)
      link editar(1,4)
      link icpp(1,5),(14,xs3)
      link editar(1,6),
     1 (10,xs3),(12,xs1)
      end
stop
```

10. CODING AN AUS MODULE

10.1. Introduction

An AUS module is similar to a stand-alone Fortran program in most respects. The distinguishing feature is the use of AUS data pools which are described in appendices A to D. Because AUS relies on the Fortran 77 OPEN statement to associate Fortran unit numbers with files, all modules of AUS must be compiled using a Fortran 77 compiler. In coding a module, maximum use should be made of these data pools so that user-supplied input is kept to an absolute minimum. All relevant data pools should be checked to ascertain whether they contain pertinent information. The AUS scheme also makes use of the operating system condition code to pass information between the module and the supervisor program. Other desirable features of an AUS module are a standard style of input data and efficient use of the available memory. Descriptions of some standard ANSTO Fortran library routines which are useful in coding a module are included in the following subsections. The AUS subroutine library, which is available as the file *aus/aus.a*, needs to be specified when the module is link edited.

10.2. Use of the OPEN Subroutine

Fortran unit numbers are assigned to files (or DD names) by passing the requirements from AUSYS to the module via a temporary file. Most modules need only include the statement

```
CALL OPEN(NOU)
```

before any input or output is performed. The variable NOU is an integer variable specifying the normal printer output unit (UNIX stdout) which may be altered by the OPEN subroutine. Of course, all output statements for the printer must use NOU for this to be effective. The main function of OPEN is to open all the required files using Fortran OPEN statements. It should be noted that files with DD names in the range from *dd17* to *dd50* are opened as unformatted and the remainder as formatted.

In modules where using OPEN to open all files is not appropriate, Fortran unit numbers greater than 50 may be used for those units not to be opened. The appropriate file name to use in opening a Fortran unit number JUNIT may be obtained using

```
CALL OPENDD(JUNIT, NAME, *N)
```

where NAME is a CHARACTER variable giving the file name (a length of 40 should suffice). The error return is taken if no specification is given for unit JUNIT.

10.3. Treatment of Character Data

10.3.1. Background

Most AUS modules were written before the introduction of character data in Fortran 77. The simplest approach to making the conversion to Fortran 77 was adopted. Therefore it was decided to retain the practice of storing character data in non-character variables throughout the bulk of the coding. Variables to be initialized with character data using a Fortran DATA statement were changed to character variables. Character data were assigned to non-character variables through a set of standard functions which use the method of writing to and reading from an internal file. These functions have proved to be completely portable. This approach to the coding led to the retention of non-character variables on the AUS data pools. The choice was then only between using two REAL variables or one DOUBLE PRECISION variable to represent some quantities. DOUBLE PRECISION has been chosen where the quantity is an obvious entity.

10.3.2. Standard Functions

Functions to 'convert' from character to non-character variables have been standardized as the following:-

CCHAR8(CVAR) returns the CHARACTER*8 variable CVAR as a DOUBLE PRECISION value,

CCHAR4(CVAR) returns the CHARACTER*4 variable CVAR as a REAL value,

ICHAR4(CVAR) returns the CHARACTER*4 variable CVAR as an INTEGER value,

CWORD8(VAR) returns the first two elements of the REAL or INTEGER array VAR as a DOUBLE PRECISION value by taking the first four characters of each element of VAR,

CWORD4(VAR) returns the first four characters of the REAL or INTEGER variable VAR as a REAL value.

10.3.3. Use of the SCAN Routine

The SCAN/VSCAN [Clancy, B. E. private communication] portable version of the SCAN free input routine includes a faithful conversion of SCAN to Fortran 77 which means that character data are returned in blocks of four byte words in non-character variables. The VSCAN entry is for use with character variables. Where 8-byte keywords are used, the CWORD8 function should be applied to the SCAN argument before a test in DOUBLE PRECISION.

The VSCAN calling sequence is the same as that of SCAN except that vector A in which input values are returned is replaced by four vectors IA, A, DA, CA which are of type INTEGER, REAL, DOUBLE PRECISION and CHARACTER respectively and are used for the return of the appropriate data type. The last argument of VSCAN which gives the number of elements has been rationalized to be the number of entities rather than the number of 4-byte words.

10.4. Use of the Condition Code

The condition code is set by a module to indicate to the supervisor whether the computation was successfully completed. The supervisor terminates the path if the condition code returned by the module is not in the range 1 to 7 inclusive. A zero condition code has been considered as an error in order to avoid the necessity of finding all error exits in existing stand-alone codes which are converted to AUS modules. To set the condition code and exit (to the supervisor), the calling sequence is

CALL CEXIT(N)

10.5. Use of Computer Memory

Modules are coded to have flexible dimensions for stored arrays and to make automatic use of the available storage. Thus, in large problems the user need only increase the region size, using the REGION environment variable, for the **aus** command.

Variable dimensions have been achieved either by using the Fortran feature, which allows variable dimensions of subroutine arguments, or by explicitly calculating all addresses within one singly subscripted variable. The latter method allows complete flexibility of use of memory but considerably increases the time required to code a module.

The VARRAY routine is used for the dynamic procurement of memory. The current version of VARRAY uses the system routines MALLOC and FREE. The four calling sequences are as follows:

(a) CALL NARRAY(N)
which returns the number of four-byte words of free memory which is determined using the REGION environment variable. Thus REGION is really used to limit the memory used by modules.

(b) CALL VARRAY(AV, IARD, N)
which procures N four-byte words of memory and returns IARD as the address of the first word of memory made available relative to the REAL vector AV and which is aligned to DOUBLE PRECISION boundary. AV may be any dimensioned variable of the subroutine which calls VARRAY.

(c) CALL VARRAY(AV, IARD, -N)
which releases the memory procured with (b).

(d) CALL RARRAY
which releases all memory made available and must be called before the module terminates.

10.6. Example of an AUS Module

The sample module listed in Appendix E has been included to illustrate some of the general features of AUS modules. The module includes sample subroutines to read the XSLIB, GEOM and FLUXB data pools as well to demonstrate most of the module requirements discussed in this section. The coding has been simplified by skipping much of the data pool information and by not making the normal tests for consistency of information.

The listing gives the complete Fortran file. After the module had been compiled and the executable stored in the *aus* directory as the file *gsredit*, it could be used with the calling sequence:

```
link gsredit(7,gm1),(9,fl2),(10,xs1)
```

For simplicity, the necessary data pools are assumed to exist already. To add a module permanently to the AUS scheme, the standard unit assignments for the module are added to the system data set *aus/linklib*.

11. REFERENCES

Barry, J.M., Pollard, J.P. [1996] - AUS diffusion neutronics module - POW3D, a mathematical description. AAEC/E612.

Bennett, N.W., Pollard, J.P. [1967] - SCAN - a free input subroutine for the IBM360. AAEC/TM399.

Clancy, B.E. [1978] - AUS module PLOTFL2. AAEC unpublished report.

Clancy, B.E. [1982] - ANAUSN - a one-dimensional multigroup SN transport theory module for the AUS reactor neutronics system. AAEC/E539.

Engle, W.W. [1967] - A user's manual for ANISN - a one dimensional discrete ordinates transport code with anisotropic scattering. K-1693.

Harrington, B.V. [1976] - AUS module AUSED - an editing program for AUS cross-section data pools. AAEC/E389.

Harrington, B.V., Pollard, J.P., Barry, J.M. [1996] - POW3D - neutron diffusion module of the AUS system, a user's manual. ANSTO/E726

Petrie, L.M., Landers, N.F. [1984] - KENO V.a - An improved Monte Carlo criticality program with supergrouping. ORNL/NUREG/CSD-2/V1.R2.

Rhodes, W.A., Child, R.L. [1988] - The DORT two-dimensional discrete ordinates code. *Nucl. Sci. Eng.* **99**, 88-89.

Robinson, G.S. [1975] - AUS - the Australian modular scheme for reactor neutronics computations. AAEC/E369.

Robinson, G.S. [1977] - AUS module MIRANDA - a data preparation code based on multiregion resonance theory. AAEC/E410.

Robinson, G.S. [1984] - Extension of the AUS reactor neutronics system for application to fusion blanket neutronics. AAEC/E583.

Robinson, G.S. [1985a] - ICPP - a collision probability module for the AUS neutronics code system. AAEC/E620.

Robinson, G.S. [1985b] - A comparison of neutron resonance absorption in thermal reactor lattices in the AUS neutronics code system with Monte Carlo calculations. AAEC/E614.

Robinson, G.S. [1986a] - EDITAR - a module for reaction rate editing and cross-section averaging within the AUS neutronics code system. AAEC/E621.

Robinson, G.S. [1986b] - CHAR and BURNMAC - burnup modules of the AUS neutronics code system. AAEC/E624.

Robinson, G.S. [1986c] - MIRANDA - a module based on multiregion resonance theory for generating cross sections within the AUS neutronics code system. AAEC/E626.

Robinson, G.S. [1987] - A guide to the AUS modular neutronics code system. AAEC/E645.

Robinson, G.S. [1993] - Generation and validation of a cross section library based on ENDF/B-VI for the AUS neutronics code system. ANSTO/E712.

Robinson, G.S. [1994] - MICBURN - module to control reactor burnup using microscopic neutron cross sections within the AUS neutronics code system. ANSTO/NTP/TN191.

APPENDIX A
AUS CROSS-SECTION DATA POOLS (XSLIB)

A1. INTRODUCTION

An XSLIB data pool is used for the storage of neutron and photon group cross-section data. The same form is used for general microscopic cross-section libraries, on which the data may be potential scattering and temperature dependent, and for macroscopic data (cm^{-1}) or microscopic data (barns) which have been derived for some particular calculation. The data pool may contain group cross sections, subgroup parameters, group scattering matrices up to any P_n order, burnup information, fission spectra, and comments on the data source. It is organised into a number of pseudo files, the first of which contains general information, and each of the rest contains information on one material. This type of structure restricts the number of groups which may reasonably be used to less than about 300. A material in the library need not be unique (*e.g.* more than one set of data may be given for an isotope) but it must be given a unique identifier. All floating point data are single precision.

The libraries *aus/endfb, aus/endf200g* and *aus/endfb6* which may be input to the MIRANDA module are XSLIB data pools of the subgroup type. These libraries are normally used as the *lib* data set on *dd31*. Other XSLIB data pools which are generated and used during a particular calculation, or a small range of calculations, are used as the *xs1*, *xs2* or *xs3* data sets. These data pools do not include subgroup data and are essentially oriented to a particular reactor or cell. Normally, they do not include a distinct treatment for photons but lump neutrons and photons together as particles. The current library structure is a modification of that originally used for neutrons only.

A2. RECORD LAYOUT

An XSLIB data pool is a Fortran unformatted file which, for NN materials, is made up of NN + 1 pseudo files. The first file contains general information on the library and each of the others contains information for a material. Each pseudo file consists of a number of Fortran records which are blocked and unblocked by a special set of input/output subroutines (Section A4). A pseudo file mark, which is the word '####', is written as the first word of the first word record of each pseudo file except the first file. That is, the '####' mark functions like an end of file mark but is written as the first word of each material file for ease of file skipping. This structure was designed for fast library access on a sequential data set.

An XSLIB may be either of sequential or direct access form. The direct form is usually used for the main cross section libraries only. For the sequential access form, the records are each 228 (single precision) words long and the '####' mark is used to locate the pseudo files. On the direct access form, the records are each 2860 (single precision) words long and an additional first record on the file of form

NN,(NREC(I),I=1,NN)

is a directory giving the starting record number for the NN material files. The first pseudo file starts at record 2 on the file. The same I/O subroutines are used for both sequential and direct access forms. The module using these routines to read a file does not need to know which form the file has.

The detailed description of the data pool is given below in terms of pseudo records but it must be remembered that these are not Fortran records and, in particular, they cannot be skipped with a short read. The character data on the data pool are stored as numeric variables rather than Fortran character variables. This is consistent with the usage which has been retained in all original AUS modules (Section 10.3). Almost all data are single precision except for some character data

which form an obvious entity. Such items are marked explicitly by *'(double precision)'*. In the description, the length of each record is given in terms of single precision words.

A3. DETAILED STRUCTURE

A3.1. First Pseudo File

A3.1.1 Library Identification

First heading record of 20 words:

words 1 and 2 = the characters **ausdata** *(double precision)*;
words 3 and 4 = the characters **nxscat** *(double precision)*;
words 5 and 6 = an 8-character library identifier *(double precision)*;
words 7 to 20 = 56 characters used to describe the data pool.

Second heading record of 10 words:

word 1 = library update identification number (integer);
word 2 = number of materials (NN);
word 3 = number of neutron groups (NG) + 1000 × number of photon groups (NGGA);
words 4 to 9 are maximum values for any material in the library of:
word 4 = number of reactions (NREAC);
word 5 = number of neutron cross-section temperatures (NXST);
word 6 = number of neutron cross-section σ_p tabulations (NSP);
word 7 = number of neutron scatter matrix temperatures (NSCATT);
word 8 = number of neutron scatter matrix σ_p tabulations (NSCSP);
word 9 = order of scatter matrix expansion (NP), see A3.1.2;
word 10 = 100 × NK + INDRES, where NK is the number of kerma factors, and INDRES takes the following values
(a) = 1, neutron cross sections are functions of potential scattering,
(b) = 2, neutron cross sections are functions of σ_0 which depends on the total cross sections,
(c) = 3, tabulated neutron 'cross sections' are actually subgroup parameters for the resonance theory of MIRANDA, used for *aus/endfb*, *aus/endf200g*, and *aus/endfb6*,
(d) = 4, cross sections and scatter matrices are tabulated for all values of LPS and KPS (see below) and this is used for irradiation-dependent data in burnup calculations.

A3.1.2 Contents

NN records of 11 words each. The N^{th} record relates to material N in the library (the N + 1 pseudo file) and the words are:

words 1 and 2 = A8 material name *(double precision)*, *e.g.* `u238`;
words 3 and 4 = A8 data source *(double precision)*, *e.g.* `endfb6`;
word 5 = A4 modification, *e.g.* `mod2`;
word 6 = ND + 100 × NAR, where ND is a number such that the highest ND for different data files for the one material is recommended data and, NAR is the number of additional reactions (default is two additional reactions if NAR is zero);
word 7 = NXST, number of neutron cross-section temperatures;

word 8 = NSP, number of neutron cross-section σ_p tabulations;
word 9 = NSCATT, number of neutron scatter matrix temperatures;
word 10 = NSCSP, number of neutron scatter matrix σ_p tabulations;
word 11 = NP which gives the order of the scatter matrix expansions and may be five hexadecimal digits each having the meaning P_n order + 1 for (in order lowest to highest digit) neutron scattering, photon production, (n,γ) photon multiplicity, fission photon multiplicity and photon interaction. Note that photon production may be split into those from (n,γ), fission and the remainder. NP may take the value zero which means no scatter matrices.

The first five words must form a unique material identifier not only on general cross-section libraries but also on a generated data pool with data obtained from, say, several cell calculations.

A3.1.3 Neutron Group Information

A record of NG + 1 words
of the lethargies of the upper (energy) boundary of the first group and the lower boundaries of all groups.

A record of NG + 1 words
of the energies in eV corresponding to the above lethargies.

A record of NG words
giving the group velocities (in 10^8 cm s^{-1}).

A3.1.4 Comments

A comment record of the form NC, (ST(K), K=1, NC)
where the words when printed (1X,20A4) form lines of comments pertaining to the general library preparation.

A3.1.5 Photon Group Information

A record of NGGA+1 words
giving energies of the photon group boundaries. Given only for NGGA non-zero.

A3.2. Material Pseudo Files

A3.2.1 File Mark

A one-word record of '####'

A3.2.2 Identification

An 11-word record
which duplicates the contents record of file 1, *e.g.* `u238, endfb6, mod2`, ND, NXST, NSP, NSCATT, NSCSP, NP.

A record of NAR words
giving the ENDFB MT numbers of the additional reactions stored on the file. If NAR is zero, the default reactions are (n,γ) and (n,2n). Some of the MT numbers used on the AUS libraries are:

16	(n,2n)		
102	(n,γ)		
103	(n,p)	203	total proton production (H gas)
105	(n,t)	205	total tritium production
107	(n,α)	207	total alpha production (He gas)
444	total damage energy		

A3.2.3 Burnup and Mass Information

A 20-word record:

words 1 and 2	=	A8 name of nuclide produced by decay *(double precision)*;
words 3 and 4	=	A8 name of nuclide produced by reaction 7 (see cross-section records below) *(double precision)*;
words 5 and 6	=	A8 name of nuclide produced by reaction 8 *(double precision)*;
word 7	=	0 to 6 to indicate fuel type (=0 if not fissionable);
word 8	=	the decay constant λ (10^{-24} s^{-1}) {A negative value is used for two decay modes and the second decay product is specified in words 5 and 6. In this case, the branching ratios for the first and second decay modes, given in words 18 and 19 respectively, are used to multiply |word 8|.};
words 9 to 14	=	fractional yield from fission of fuel of type 1 to 6;
word 15	=	fission energy release (joule/fission);
word 16	=	atomic mass (^{12}C scale) of an isotope;
word 17	=	may be positive, zero or negative –
	(a)	$> 0.$, atomic mass of second isotope of the nuclide which is a molecule,
	(b)	$= 0.$, implies one isotope only,
	(c)	$< 0.$, -D is given, where D is the average spacing between resonances (in eV);
word 18	=	fraction of potential scattering due to the first isotope;
word 19	=	fraction of potential scattering due to the second isotope;
word 20	=	NB, the burnup order indicator.

NB is used to re-order isotopes before analytic solution of the depletion equations. With few exceptions Robinson [1986b], NB is a 7 decimal digit fixed point word PCCAAAM, where

P	=	1 for a fission product, else 0,
CC	=	the charge number,
AAA	=	the mass number,
M	=	1 for the ground state when a metastable isomer is also given, else 0.

The nuclide names given in words 1 to 6 inclusive need not be in the library. If word 1 is set to the three characters NO0, the nuclide does not burn out by decay. If both word 3 and word 5 are NO0, the nuclide does not burn out by neutron absorption.

A3.2.4 Fission Spectrum

A record of NG words
giving the fission spectrum normalised to unit sum.

A3.2.5 Temperature and σ_p Tables

A record of NXST words
giving the temperatures (K) at which neutron cross sections are tabulated. The temperatures are monotonic decreasing.

A record of NSP words
giving either
(a) set of σ_p (or σ_0) in decreasing order for which neutron cross sections are tabulated, or
(b) for INDRES=3, the number 1.E+20 and (NSP-1) subgroup values of σ_{tot} in increasing order.
The subgroup theory assumes that group resonance integrals, RI, can be obtained from

$$RI/\delta u = \sum_{i=1}^{NSP-1} \frac{w_i s}{s + \sigma_{tot}^i}$$

where s is related to σ_p.

A record of NSCATT words
giving the temperatures (K) at which neutron scatter matrices are tabulated. The temperatures are monotonic decreasing.

A record of NSCSP words
giving the set of σ_p (or σ_0) in decreasing order for which neutron scatter matrices are tabulated.

A3.2.6 Neutron Cross-section and Scatter Matrix Data

The data in this block are given for each group in turn. For any group the data are given as described in the following paragraphs.

A3.2.6.1 Cross Sections

A set of records each of the form LPS, LV, (XS(K), K=1,LV):

LPS = -1 implies the set consists of this one record only, *i.e.* the cross sections are not tabulated;
LPS = -2 implies NXST × NSP records are given, with records for NXST temperatures being given in turn for each σ_p value;
LV = length of the XS vector and not all reactions need be given. The value of LV for a material is constant for all cross-section records but may be a different constant for subgroup parameter records.

The set of vectors XS is a set of cross-section records if INDRES ≠ 3 but, for INDRES = 3, the first NXST records are cross-section records and the remainder are subgroup parameter records.

For ***cross-section records***, the data are

XS(1) = σ_{tr}, the transport cross section;
XS(2) = σ_{abs}, the absorption cross section;
XS(3) = $\nu\sigma_f$, the fission emission cross section;
XS(4) = σ_s, the scattering cross section, if LPS = -1, or σ_p, the potential scattering cross section, if LPS = -2;
XS(5) = σ_{tot}, the total cross section;
XS(6) = σ_f, the fission cross section;
XS(7) to XS(NX) = the cross sections for the NAR additional reactions (NX = NAR +6), and the first two cross sections given may be used as the first and second burnup reactions;
XS(NX+1) = total kerma factor excluding following entries;
XS(NX+2) = elastic scattering kerma factor;
XS(NX+3) = (n,γ) kerma factor;
XS(NX+4) = fission kerma factor.

The kerma factor is the cross section times the energy deposited in the units of barn eV. To allow for resonance shielding, elastic scattering, capture and fission may have separate entries and the total kerma factor is obtained by summing the values given. For any given nuclide, not all NK values need be given, *i.e.* LV may be less than NX+NK (=NXK).

Additional entries may be added depending on the library type:
(a) For macroscopic materials, additional parameters may be

XS(1+NXK) = radial diffusion coefficient,
XS(2+NXK) = axial diffusion coefficient,

or for 3D calculations XS(1+NXK), XS(2+NXK) and XS(3 +NXK) may be x, y, z diffusion coefficients.

(b) For resonance groups on an INDRES=3 library

(i) XS(1+NXK) = average peak resonance height σ_0;
XS(2+NXK) = average value of $2E_r/\Gamma$ where E_r is resonance energy and Γ is total width; or
(ii) XS(1+NXK) = XS(2+NXK) = 0 implies narrow resonance theory; or
(iii) XS(1+NXK) = 1., XS(2+NXK) = 0 implies very wide resonance extending over many groups; or
(iv) XS(1+NXK) = $-\sigma_{inel}$, XS(2+NXK) = 0 implies narrow resonance theory and inelastic scattering; or
(v) XS(1+NXK) = $-\sigma_{inel}$, XS(2+NXK) = 1. implies very wide resonance and inelastic scattering; additionally
(vi) XS(3+NXK) may be the ratio of isotropic to anisotropic P_0 removals;
(vii) XS(4+NXK) may be the position factor (see Section 4. of MIRANDA documentation) for absorption within the group; and
(viii) XS(5+NXK) may be the position factor for fission within the group.

For ***subgroup parameter*** records the data are

XS(1) = subgroup weight for absorption, w_i,
XS(2) = subgroup weight for resonance scattering,
XS(3) = subgroup weight for fission,
XS(4) = subgroup weight for fission emission,
XS(5) = subgroup weight for the ratio of group flux to asymptotic group flux, but XS(3) and XS(4) may be omitted so that XS(3) becomes the subgroup weight for flux ratio.

A3.2.6.2 Scatter Matrices

For each P_n scattering order, the following data are given in turn unless NP=0 when no data are given.

A set of records each of the form KPS, KV, (SCAT(K), K=1, KV):

KPS = position of the self-scatter term in the SCAT vector but has the additional implications
(a) KPS > 1, the set consists of NSCATT records giving temperature-dependent thermal data;
(b) KPS = 1, the set consists of this one record;
(c) KPS = 0, the set consists of NSCATT × NSP records with NSCATT records being given in turn for each σ_p value. The position of self scatter is 1.

These additional implications apply to the first record of a set only and for the following records, if any, KPS is simply the self-scatter position.

KV = length of the SCAT vector, $1 \leq KV \leq NG$;
SCAT = a vector of outscatter P_n components, $\sigma^n_{g\to g'}$, from the current group, g.

NOTES

(1) $\sigma^{n}_{g\to g'} = \int_{-1}^{1} \sigma_{g\to g'}(\mu)\, P_n(\mu) d\mu$,

where μ is the cosine of the scattering angle.

(2) The self-scatter term of the P_0 vector is a 'true' self-scatter term if NP > 1, but it is transport-corrected if NP = 1. In any module using a P_0 calculation, the library self-scatter term should be ignored. Neutron balance should be conserved by using σ_{tr} (reaction 1) and the relation

$\sigma_{g\to g} = \sigma_{tr} - \sigma_{abs} - \sum_{g'\neq g} \sigma_{g\to g'}$,

In higher order P_n calculations, the P_0 self-scatter term on the library should be used and the total cross section obtained not from reaction 5 but from

$\sigma_{tot} = \sigma_{abs} + \sum_{g'} \sigma_{g\to g'}$.

Any group condensation must be consistent with these two conventions.

(3) The (n,2n) reaction also requires special consideration. The standard convention is adopted with 2 $\sigma^{(n,2n)}_{g\to g'}$ being included in the scatter vector and $\sigma_{n,2n} = \sum_{g'} \sigma^{(n,2n)}_{g\to g'}$ being subtracted from cross section 2, *i.e.*

$\sigma_{abs} = \sigma_{cap} + \sigma_f - \sigma_{n,2n}$

where σ_{cap} includes all reactions which do not result in neutron emission.

For (n,3n) reactions, 3 $\sigma^{(n,3n)}_{g\to g'}$ is included in the scatter vector and $2\sigma_{n,3n}$ is subtracted from σ_{abs}.

This is the end of the neutron group data description.

A3.2.7 Material Comments

A comment record of the form NC, (ST(K), K=1, NC)
where the words when printed (1X,20A4) form lines of comments pertaining to this material.

A3.2.8 Photon Production Matrix Data

To allow for resonance shielding, photon production data for (n,γ) and fission reactions may be given separately as multiplicity matrices. That is, the photon production matrix is obtained by multiplying the given data by the appropriate group cross section. For each neutron group in turn, data are given for

Set 1 - the P_n expansion of photon production excluding that given in Set 2 or Set 3,
Set 2 - the P_n expansion of the (n,γ) multiplicity, and
Set 3 - the P_n expansion of the fission multiplicity.

The P_n expansion order for each set is extracted from NP. For each value of n in a set, the given record has the form

MPS, MV, (VECT(K), K=1, MV)

MPS = 1000 x set number + first photon group in which photons are produced,
MV = the number of groups in which photons are produced, and
VECT = a vector giving the P_n components of the photon production (or multiplicity).

A3.2.9 Photon Interaction Data

No data are given in this block if the photon interaction scattering order, which is taken from NP, is zero. The data given for each photon group in turn are similar to those given for neutron cross sections which are independent of σ_p and T. The cross section record is

-1, LVA, (XS(K), K=1, LVA)

XS(1) = transport cross section,
XS(2) = absorption cross section,
XS(3) = zero,
XS(4) = Compton scattering cross section,
XS(5) = total cross section,
XS(6) = zero,
XS(7) = photo-electric cross section,
XS(8) = pair production cross section, and
XS(9) = kerma factor

The records in the P_n scattering expansion are

LPS, LV, (SCAT (K),K=1,LV)

LPS = the position of self-scatter,
LV = the number of outscatters from the photon group,
SCAT = the P_n components of the photon group outscatters.

This completes a material file.

A4. INPUT/OUTPUT ROUTINES

A set of subroutines is provided for blocking and unblocking the records on an XSLIB data pool. These routines should be used for all access to the data pools. As the input and output routines are separate and use different buffers, one data set may be written while another is being read. For special applications requiring the simultaneous reading of two data pools, a second set of read routines is provided which function identically to those described but they have a "2" added to the name of each entry, *e.g.* ARDP2.

A4.1. Input Routine

There are four entries to the subroutine, namely ARDP, ARDN, ARDS, and ARDT. A description of the calling sequences follows.

(a) The library positioning entry is

CALL ARDP (IT, K)

where IT is the Fortran unit number,
K=0 implies rewind and must be given to open a data set, and
K>0 skips K pseudo file marks.

This entry must be used to skip a pseudo file mark, even if it is the next word.

(b) The N-word read entry

CALL ARDN (ST, N)

reads |N| words and returns the words in the vector ST if N is positive. A negative value of N is used for skipping unwanted data and a nil result is obtained from N=0.

This entry is used for file 1 data and material file data except cross-section (or sub-group parameter) and scatter matrix records. Note also that CALL ARDN (ST, I) and CALL ARDN (ST(I + 1), J) are equivalent to CALL ARDN (ST, I + J). A DOUBLE PRECISION variable may be read or written using for example:

CALL ARDN(DWA,2)

(c) The cross-section and scatter vector read entry

CALL ARDS (LPS, LV, ST)

reads the next pseudo record and returns LPS = first word, LV = second word and (ST(K), K=1,LV).

The cross-section (or subgroup parameter) and scatter vector pseudo records must be read with this entry and not ARDN. There is a lapse of correspondence with the actual data pool pseudo records if NP=0. In this case, a P_0 scatter vector is returned with LPS=1, LV=1, ST(1)= $\sigma_{tr} - \sigma_{abs}$.

(d) The library contraction entry is

CALL ARDT (MNXST, MNSP, MNSCAT, MNSCSP, MNP)

This entry is used if the programmer wishes to treat an XSLIB data pool as if it were a subset of the actual data set. The input parameters in the argument list correspond with the five maximum values of the words 5 to 9 of the second data pool heading record. If the parameter is greater than one, the corresponding set of data is all returned by the ARDS entry. The additional meanings are

MNXST=MNSP=1	the cross-section record for the highest σ_p value and the lowest temperature is the only one returned,
MNSCAT=1	the scatter records for the lowest temperature only are returned,
MNSCSP=1	the scatter records for the highest σ_p value only are returned (with KPS=1 if the library has KPS=0),
MNP=1	only the P_0 scatter vectors are returned,
MNXST=MNSP=0	no cross-section records are returned, and
MNSCAT=MNSCSP=MNP=0	no scatter records are returned.

Library contraction may not be used if any data following the neutron cross section and scattering data are required. A library containing separate neutron and photon data may be read as if it contained only neutron data provided that the number of groups NG is used modulo 1000, and the scattering order NP is used modulo 16.

A4.2. Output Routine

There are four entries to the routine namely AWRD, AWRP, AWRN and AWRS.

(a) The output is a sequential file unless the following entry to specify direct access mode is called before the AWRP entry:

CALL AWRD (TRUE)

where TRUE is a logical variable with value true.

(b) The library positioning entry is

CALL AWRP (IT, K)

where IT is the Fortran unit number,

K=0 implies rewind and must be given to open the data set,
K>0 writes a pseudo file mark, and
K<0 writes an actual file mark.

(c) The N-word write entry

CALL AWRN (ST, N)

writes N words of the vector ST.

(d) The cross-section and scatter vector write entry

CALL AWRS (LPS, LV, ST)

writes the pseudo record LPS, LV, (ST(K), K=1, LV).

Example

The sample module given in Appendix E includes a subroutine RDXS which might be used to read a simple XSLIB data pool of neutron data without tabulated cross sections. In this sample read subroutine, only the essential data have been read and the rest have been skipped. Only the data σ_{tr}, σ_a, $\nu\sigma_f$ and $\sigma_{g\to g'}$ are returned from the subroutine. Note the use of CALL ARDT to ensure that more general data pools, particularly those with higher order P_n scattering data, can be read.

APPENDIX B
AUS GEOMETRY DATA POOLS (GEOM)

B1. INTRODUCTION

A GEOM data pool contains information on the geometry of some reactor system or component. The data include geometry type, mesh intervals and material layout. The material layout is in terms of materials which are specified by number only, with the numbers corresponding to position on some associated cross-section data pool. Almost all data are single precision except for some character data which form an obvious entity. Such items are marked explicitly by *(double precision)*. In the description, the length of each record is given in terms of single precision words.

The GEOM data pool is normally used as the *gm1* data set on *dd35*. It is a sequential unformatted data set.

The sample module of Appendix E includes a subroutine RDGEOM for reading a GEOM data pool describing a one-dimensional geometry.

B2. DETAILED STRUCTURE

Heading record of 21 words:

words 1 and 2	=	an 8-character geometry identifier *(double precision)*;
words 3 to 18	=	a 64-character heading,
word 19	=	the number of dimensions described (ND),
word 20	=	the geometry type (IGEOM),
where IGEOM	= 0	for rectangular geometry xyz,
	= 1	for cylindrical geometry rz,
	= 2	for spherical geometry r, and
word 21	=	NLS is 1 normally, but 5 for cluster geometry and the number of axial mesh intervals in 3D geometry. Then 2 + ND + NLS is the number of records describing a geometry.

Mesh interval record of form NXMI, (XM(I), I=1, NXMI) is given if ND>0

where XM are the mesh interval widths in cm, and
NXMI is the number of mesh intervals for the first dimension (that is, the x or r direction).

A further mesh interval record of the same form is given for each additional dimension. Thus there is a total of ND mesh interval records.

Boundary condition record of 2 × ND words:

word 1	=	left boundary condition of first dimension,
word 2	=	right boundary condition of first dimension, and
words 3 to 6,		where required, are for the second and third dimensions.

A one word dummy record is given if ND=0.

The boundary condition normally has the following meanings:

- <0. implies periodic,
- =0. implies reflective, and
- >0. implies free and is used by diffusion codes as the extrapolation distance in transport mean free paths (usual value is 0.71).

For cylindrical geometry, additional interpretations are

word 1 = number of sides of a polygonal outer boundary, and a zero value implies circular; and

word 2 = -(number of mesh intervals to be treated as the inner region in applying a polygonal reflective boundary condition limited to two regions). Zero and positive have the normal meanings.

Both numbers are given in floating point. In cluster geometry, these boundary conditions are applied to the pin-cell boundaries and the outer boundary is taken as circular, white reflective.

Material layout record of the form NL, (LAYOUT(I), I=1, NL)

NL = the product of the number of intervals in each dimension, excluding any third dimension,

LAYOUT = the array of material numbers (LAYOUT(IX,IY) in the 2D case), where the number corresponds to the material order on an XSLIB data pool, and the record is repeated for each XY plane in 3D geometry.

This completes the geometry description except for a cluster geometry which requires the following additional records.

A record of the form NL1, NPITCH, NTYPE, (NROD(I), MROD(I), PROD(I), QROD(I), I=1, NPITCH)

NL1 = $4 \times \text{NPITCH} + 2$,

NPITCH = number of rings of rods,

NTYPE = 0,

NROD(I) = number of rods equally spaced on the ring I, numbered from the centre outward,

MROD(I) = number of subdivisions of a rod on ring I,

PROD(I) = pitch radius of the ring I, and

QROD(I) = angular displacement in radians of one of the rods of the ring I from a reference diameter of the cluster.

A record of the form NL2, ((DR(I,J), I=1, MROD(J)), J=1, NPITCH)

NL2 = $\sum \text{MROD}(J)$,

DR = mesh interval of the radial subdivisions of a rod, in cm.

A record of the form NL2, ((LAYOUT(I,J), I=1, MROD(J)), J=1, NPITCH)

LAYOUT = material numbers corresponding to the radial subdivision of the rods.

A record of the form NL3, (VOL(I), I=1, NL3)

NL3 = NXMI + NL2, and

VOL = region volumes with the volumes of the main annuli being given first, followed by rod subdivision volumes in the same order as the previous records.

This completes the description of the geometry. Other sets of geometry data may follow but this feature is supported only by the POW3D module.

APPENDIX C
AUS FLUX DATA POOLS (FLUXA AND FLUXB)

C1. INTRODUCTION

Unfortunately, a general data pool to store neutron group fluxes has not been incorporated in the AUS scheme. Use is made of a POW-type flux dump as a FLUXA data pool and of a WDSN-type flux dump as a FLUXB data pool. The FLUXA data pool is used for 1D or 2D edge mesh fluxes and the FLUXB data pool is used for 1D mesh average fluxes. An extension of the FLUXA data pool is used for 3D edge mesh fluxes.

The FLUXA data pool is normally used as the *fl1* data set on *dd36* and the FLUXB data pool as the *fl2* data set on *dd37*. Both are sequential unformatted data sets. Items given in double precision are marked explicitly by *(double precision)*.

The sample module in Appendix E includes a sample subroutine RDFLB for reading a FLUXB data pool.

C2. DETAILED STRUCTURE OF A FLUXA DATA POOL

First record of 33 words:

words 1 and 2	=	an 8-character flux identifier *(double precision)*,
words 3 to 18	=	a 64-character heading,
words 19 and 20	=	the characters **real** or **adjoint** *(double precision)*,
words 21 and 22	=	the characters **eigenv** or **source** or **kinetics** *(double precision)*,
words 23 and 24	=	the characters **problem** or anything else *(double precision)*,
word 25	=	0. or, for kinetics calculation, the power or flux,
word 26	=	0. or, for kinetics calculation, the time in seconds,
word 27	=	number of outers or, for kinetics calculations, the time step number,
word 28	=	MAXX, the dimensioned size of the X mesh,
word 29	=	MAXY, the dimensioned size of the Y mesh,
word 30	=	NGIGD, the number of energy groups (plus the number of delayed groups if a kinetics calculation),
word 31	=	NXM, number of X mesh points, *i.e.* number of mesh intervals plus 1,
word 32	=	NYM, the number of Y mesh points, and
word 33	=	NG, the number of energy groups.

Second record of the form AKEFF, AEIGEN, BKEFF, BEIGEN, MOP, DLAM, RACCFO, SOMEGA, (OMEGA(I), I=1, NGIGD), (((FLX(I,J,K), I=1, MAXX), J=1, MAXY), K=1, NGIGD),
where the floating point variables are single precision, except for FLX which is double precision, and

AKEFF	=	the multiplication constant k_{eff} corresponding to λ_1;
AEIGEN	=	λ_1, a criticality search eigenvalue or 1;
BKEFF	=	the k_{eff} corresponding to λ_2;
BEIGEN	=	λ_2, a second criticality search eigenvalue or 1;
MOP	=	convergence stage and takes the values 3 means calculation just started, 2 means outer extrapolation accepted hesitantly, 1 means outer extrapolation accepted readily, 0 means converged;

DLAM = dominance ratio for outer iteration (ratio of second biggest to biggest eigenvalue);
RACCFO = accuracy of last outer;
SOMEGA = trial value for inner SLOR coefficients, usually 1;
OMEGA = vector of inner SLOR coefficients for each energy group; and
FLX = double precision array of edge fluxes (plus volume weighted precursor concentrations for kinetics calculations).

As already stated, this is a POW-type flux dump and, from the second record, only AKEFF, MOP (converged or not) and FLX have a meaning for other modules beside POW3D. Additional 2-record flux dumps may be included in the one data set.

C3. DETAILED STRUCTURE OF A 3D FLUXA DATA POOL

First record of 35 words:

words 1 to 29 = as for the 1D and 2D FLUXA data pool (Section C2),
word 30 = MAXZ, the dimensioned size of the Z mesh,
word 31 = NGIGD, the number of energy groups (plus the number of delayed groups if a kinetics calculation),
word 32 = NXM, number of X mesh points, *i.e.* number of mesh intervals plus 1,
word 33 = NYM, the number of Y mesh points, and
word 34 = NZM, the number of Z mesh points, and
word 35 = NG, the number of energy groups.

Second record of the form AKEFF, AEIGEN, BKEFF, BEIGEN, MOP, DLAM, RACCFO

where all variables have the same meaning as for the 1D and 2D FLUXA data pool.

A set of NZM × NGIGD records for each Z mesh point for each neutron (and delayed neutron group) of the form

((FLX(I,J), I=1,MAXX), J=1,MAXY)

where FLX is a double precision array as for the 1D or 2D FLUXA data pool, and NZM records are given for each group in turn.

C4. DETAILED STRUCTURE OF A FLUXB DATA POOL

The data pool consists of a number of pseudo files which are separated by the single word record '####'. The records of each file are as follows:

First record of six fixed point words:

word 1 = an indicator which takes the values
1 normal case, fully converged,
2 normal case, not converged,
5 normal case, fully converged, adjoint may also be given,
6 adjoint case, fully converged,
7 normal case, not converged, adjoint may also be given,
8 adjoint case, not converged;
word 2 = the geometry type, taking the values
-1 cylinder,
0 slab,
1 sphere;
word 3 = product of the number of groups and the number of mesh intervals;

word 4 = 0;
word 5 = (2n - 1) where n is the P_n order of scattering; and
word 6 = 0.

Heading record of 18 words:

words 1 and 2 = an 8-character flux identifier *(double precision)*;
words 3 to 18 = a 64-character heading.

A record of EIGEN, NG, NR, (FLX(I,J), I=1,NR), J=1,NG)

where EIGEN = $1/k_{eff}$ or total activity for source calculations (*double precision*);
NG = number of energy groups;
NR = number of mesh intervals; and
FLX = the mesh average scalar flux (*double precision*).

The ANAUSN module writes angular flux starters at the end of this record to assist restarts, but accepts input flux guesses on which the number of starters is inappropriate.

A set of m records of the form (FLX(I,J,L), I=1,NR), J=1,NG)

giving the currents for P_n scattering calculations with n>0, where m=n except in cylindrical geometry for which m = (n×(n+4))/4. The data are double precision.

APPENDIX D
AUS STATUS DATA POOLS

D1. INTRODUCTION

STATUS data pools are Fortran sequential, unformatted data sets which are normally used as the pair of data sets *st1* and *st2* on *dd38* and *dd39* of the AUS system, respectively. The two STATUS data pools are used in combination, with *st2* acting as a pointer to the main *st1* data set. The data pools consist of a sequence of entries with each entry determining its own function. The form of the entries is fixed, however, so that a module may skip or copy entries without knowing the details of all entries. Each module of a calculation may add additional entries to the end of the data pool by using the sequence: search for end of file, backspace and write. That is, the data pools are 'add-on' data sets in which all entries are retained throughout a calculation sequence.

The data pool contains entries for isotopic compositions, spatial smearing factors and other miscellaneous data which together form a history of the functions that have been performed by the modules during a sequence of calculations. Its major purpose is to allow the automatic evaluation of nuclide reaction rates at any stage of a calculation. The data pool differs from other AUS data pools in several ways; it can embrace an entire calculation sequence, it can include general information, and it can take part in controlling the AUS calculation. Because of this generality, the use that each module makes of the data pool must be carefully defined and suitable labelling conventions must be established to differentiate between data generated within different subsections of the overall calculation. To be more specific, we wish to be able to calculate nuclide reaction rates in the components of a lattice cell which, together with other cells, forms a representation of a reactor core.

D2. DATA POOL CONTENTS IN DETAIL

D2.1. Entry Format

Each entry consists of two or more records, the first of which establishes the type of entry. This first 7-word record is

(A(I), I=1,5),N,M

where A is either a fully qualified material name of 20 characters or it has the form

A(1 - 2)	=	**$data**,
A(3 - 4)	=	the name of module writing the entry,
A(5)	=	the entry type, *e.g.* **time**, **cell**, **grps**, and
|N|	=	the number of groups of information in the following records.

If N is negative, each information group begins on a new-record thus enabling large quantities of data to be included easily. M is the maximum size of an information group in single precision words.

The trailing records have the form

((B(I,K), I=1,M), K=1,N) – one record for N positive, or
J,(B(I), I=1,J) – repeated |N| times for N negative.

D2.2. Mixing Rule Entry

This is the basic entry in the data pool and it determines the entry format. Mixing rules describe either the mixing of nuclides to form a discrete material or the spatial smearing of materials. The name of any nuclide or material consists of 20 bytes as in the XSLIB data pool. The conventions

for naming are given in section D3.2.

The entry is (for positive N)

(A(I), I=1,5),N,M+5
((B(I,K), I=1,5), (C(I,K), I=1,M), K=1,N)

A = the name of a material formed from N constituents,
B = the set of constituent names, and
C = (a) the concentrations in atoms per barn cm of each nuclide, for M=1, or
(b) the spatial smearing factors for each material f_i, d_{ig} for M>1, *i.e.* M is usually one more than the number of groups before energy condensation, but is one more than the number of groups after condensation if the nuclide data are condensed. In both cases d_{ig} is given for the same number of groups as the nuclide cross-section data pool.

The factors f_i, d_{ig} are given by

$$f_i = V_i / V_I \quad \text{, and}$$

$$d_{ig} = V_I \, \phi_{ig} / (\sum_{g \in G} \sum_{i \in I} V_i \phi_{ig}) \quad \text{, or}$$

$$d_{iG} = V_I \sum_{g \in G} \phi_{ig} / (\sum_{g \in G} \sum_{i \in I} V_i \phi_{ig})$$

where V_i and ϕ_{ig} are region volumes and fluxes which are used to group condense and smear into group G and region I.

D2.3 The prog Entry

The entry is

DATA, MODNAM, PROG, 1, 1
NPROG

DATA = the A8 character data $**data**,
MODNAM = the A8 module name, and the other entry types below are similar,
PROG = the A4 character data **prog**,
NPROG = the number of the current operating program or operating cycle.

D2.4. The time Entry

The entry is

DATA, MODNAM, TIME, 1, 3
TNOW, TLAST, PTGRAL

TIME = the A4 character data **time**,
TNOW = the current time in days,
TLAST = the previous time in days, and
PTGRAL = the integral of power of the total system with respect to time in watt days or watt days cm^{-3}.

D2.5. The irad Entry

The entry is

DATA, MODNAM, IRAD, N, M
((A(I,J), I=1,5), B(J), VOL(J), (FLUXT(K,J), K=1,NFLX), J=1,N)

IRAD = the A4 character data **irad**,
A = the set of names of discrete materials which are burnt up,
B = the integral of power density with respect to time in watt days cm^{-3} for each material,
VOL = the volume occupied by the material in cm^3,
FLUXT = the fluence, or time integral of a detector reaction rate, for the material,
NFLX = the number of FLUXT values, M - 7, which may be zero.

D2.6. The cell Entry

The entry is

DATA, MODNAM, CELL, 1, 3
CNAM, NCELL

CELL = the A4 character data **cell**,
CNAM = the A8 name of the cell,
NCELL = a count of cell calculations for the current time step.

D2.7. The grps Entry

The entry is

DATA, MODNAM, GRPS, 1, N
(IGB(I), I=1,N)

GRPS = the A4 character data **grps**,
N = the number of condensed groups plus one,
IGB = a set of fixed point numbers giving the first group of condensed group 1, and the last group of each condensed group. The numbers are in terms of the previous group set.

D2.8. The gfac Entry

The entry is

DATA, MODNAM, GFAC, 1, N
(A(I), I=1,NG)

GFAC = the A4 character data **gfac**,
NG = the number of energy groups; and
A(I) = the ratio of the k_{eff} flux for a cell calculation to the k_{∞} flux, for each group I.

D3. NOTES ON USAGE

D3.1. General Comments

The *st1* data set is the main data pool to which all entries should be added. The *st2* data set serves mainly as a pointer to the data in *st1* and, as such, contains the **prog**, **time** and **cell** entries which give structure to the data pool.

A **prog** entry is only required if the burnup is divided into a number of programs (*i.e.* operating cycles) for each or which the time is considered to restart from zero. One **prog** entry is given for each cycle and precedes all other entries for that cycle. One **time** entry is given for each time step (including time zero) and precedes all other entries, except **prog**, for that time. A single **irad** entry follows each **time** entry, except possibly the first. A **cell** entry is given for each lattice cell calculation and immediately precedes entries written for that cell by a cross-section generation module.

Each set of mixing rule entries immediately follows an appropriate **grps** entry to define the group condensation. The mixing rules must be entered such that the order of materials on the STATUS data pool is the same as that on the cross section data pool. A further requirement is that the order of constituents within a spatial smearing rule be the same as the order of the definitions of those constituents on the STATUS data pool.

The **gfac** entry which enables burnup of a cell in a critical spectrum is exceptional because it is written on the *st2* data set as well as on *st1*.

The use of STATUS data pools requires that materials be divided into two classes called materials and nuclides. A nuclide is present on the main cross-section library of the cross-section generation module and has microscopic cross sections. It appears only on the right hand side of a mixing rule. A material is composed of nuclides and must be defined by a mixing rule. It may be macroscopic or microscopic and may even be identical in cross section to a nuclide. Materials and nuclides may be in the same cross-section data pool, but usually they are in different data pools which may have a different number of groups.

D3.2. Material Names

The materials and nuclides in an AUS calculation all have 20-byte names which are used to provide a unique identification. These names are constructed using the set of conventions detailed below.

The first requirement is that the name of a material or nuclide must be exactly the same on the STATUS and cross section data pools. The 20 bytes consist of two double precision words each containing 8 bytes of information and one single precision word with 4 bytes. The first word gives the simple name for a nuclide (*e.g.* `u235`) or the name of a material supplied by the user in defining that material. The second word gives the name of the cell calculation in which the cross sections were generated and, where necessary, a number to indicate the region of the cell to which the cross sections apply. The user-supplied cell calculation name must be different for different cells within the one system and should be restricted to six characters to allow for the cell region number. The last word is modified for a material (but not a nuclide) each time the material is condensed. These four characters go through the sequence `orig`, `mod1`, `mod2`, *etc.*

D3.3. Functions of the AUSYS Supervisor Program

The initialisation and simple editing features required for STATUS data pools are supplied by the AUSYS program. The *st1* data set is often saved from one AUS run to another. The *st2* data set is not normally saved; it is initialised by AUSYS at the start of an AUS calculation by copying **prog**, **time**, **cell** and **gfac** entries from *st1* to *st2*. If *st1* is new and is therefore empty, this results in *st2* being empty also.

Entries to modify this standard option may be included in the AUSYS input stream following the **step** *named* or **stop** directives. The data set is identified by the entry

$dd*nn* **disp=new**
or
$dd*nn* **disp=old**

where **dd***nn* is the DD name of the data set,
disp=new causes an end of file to be written at the beginning,
disp=old results in no action.

The identifying entry may be followed by an entry requesting an end of file to be inserted before a nominated entry. This assists recovery following an error. The entry has the form

$eof [**prog** = *nprog*] **time** = *time* [**cell** = *cell-name*]

The specification of the program (cycle) number, *nprog*, is not required for a single cycle burnup calculation. If **cell** = *cell-name* is not given, the end of file is inserted before the nominated **time** entry. Otherwise, it is inserted before the **cell** entry for the cell-name for the nominated time (for the nominated cycle). A negative value of *time* is used as a counter. Thus -2 causes an end of file before the second **time** entry.

Entries to be added at the end of a data set may be included after a **$dd***nn* or **$eof** entry. The lay-out of these free format entries is exactly the same as that of an entry in the data pool. However, no notice is taken of end of records. All 20-byte alphanumeric names must be given as two words of length 1 to 8 characters and one word of length 1 to 4 characters. Special characters or blanks may be used to separate information. The layout of the entries is compatible with the card image output produced by the PSTAT subroutine of AUSYS. A **$dd99** entry is used to terminate the STATUS data entries if other AUSYS input is provided.

To restart a normal lattice burnup calculation which is correct as far as it has gone, the user does not need to supply STATUS input to restart immediately before the CHAR module. To restart after the CHAR module, the following input is required:

$dd39 disp=new
$data miranda time 1 3 *time* **0. 0.**

where *time* is the current burnup time.

APPENDIX E
A SAMPLE MODULE

```
C          main routine of a sample AUS module which does some simple
C          editing following a 1 dimensional flux calculation
C
      COMMON NG,NN,NMESH,NGD,NND,NMD,A(2)
C          open AUS files,  allow NOU to be changed
      CALL OPEN(NOU)
C          obtain no. of words(*4) of storage available
      CALL NARRAY(NARD)
C          allow room for buffers,etc
      NARD=NARD-5000
C          request NARD words of storage
C          IARD is returned as address of first word of storage
C          relative to *4 vector s
      CALL VARRAY(A,IARD,NARD)
C          set array sizes. these would be obtained in practice by
C          a preliminary read of data pools
      NGD=50
      NND=20
      NMD=50
      NXS=NGD*NND
C          assign storage
      LFLUX=IARD
      LXM=LFLUX+NMD*NGD*2
      LVOL=LXM+NMD
      LAY=LVOL+NMD
      LTR=LAY+NMD
      LABS=LTR+NXS
      LANUF=LABS+NXS
      LSCAT=LANUF+NXS
      LANS=LSCAT+NGD*NGD*NND
      LAANS=LANS+NND*NGD*2
      LMAX=LAANS+NND*2
      IF(LMAX.LE.IARD+NARD)GO TO 3
      WRITE(NOU,2)
    2 FORMAT(' STORAGE EXCEEDED')
C          note error exits do not need condition code set
      CALL VEXIT(0)
C          call calculation subroutine
    3 CALL EDIT(NOU,A(LFLUX),A(LXM),A(LVOL),A(LAY),A(LTR),A(LABS),
     1 A(LANUF),A(LSCAT),A(LANS),A(LAANS))
      STOP
      END
C
C
      SUBROUTINE EDIT(NOU,FLUX,XM,VOL,LAY,TR,ABS,ANUF,SCAT,ANS,AANS)
      COMMON NG,NN,NMESH,NGD,NND,NMD,A(2)
      DOUBLE PRECISION FLUX(NMD,NGD)
      DIMENSION XM(NMD),VOL(NMD),LAY(NMD),TR(NGD,NND),ABS(NGD,NND),
```

```
     1 ANUF(NGD,NND),SCAT(NGD,NGD,NND),ANS(NND,NGD,2),AANS(NND,2)
       DOUBLE PRECISION HEAD(2),CCHAR8
C           illustrates use of CCHAR8 to set HEAD to character data
       HEAD(1)=CCHAR8('NU-FISS ')
       HEAD(2)=CCHAR8('ABS.    ')
C           read geom data pool
       CALL RDGEOM(7,NN,NMESH,XM,VOL,LAY)
C           read fluxb data pool
       CALL RDFLB( 9,NMD,NFM,NG,FLUX)
C           read xslib data pool
       CALL RDXS(10,NGD,NN,NGX,TR,ABS,ANUF,SCAT)
C           edit of nu-fiss and abs by material and group
       DO 3 IG=1,NG
       DO 1 I=1,NN
       ANS(I,IG,1)=0.
     1 ANS(I,IG,2)=0.
       DO 2 IX=1,NMESH
       IM=LAY(IX)
       VF=VOL(IX)*FLUX(IX,IG)
       ANS(IM,IG,1)=ANS(IM,IG,1)+ANUF(IG,IM)*VF
     2 ANS(IM,IG,2)=ANS(IM,IG,2)+ABS(IG,IM)*VF
     3 CONTINUE
       DO 8 J=1,2
       WRITE(NOU,4)HEAD(J),(I,I=1,NN)
     4 FORMAT('0',A8,' BY MATERIAL AND GROUP'/1X,10I12)
       DO 5 I=1,NN
     5 AANS(I,J)=0.
       DO 7 IG=1,NG
       WRITE(NOU,6)IG,(ANS(IM,IG,J),IM=1,NN)
     6 FORMAT(1X,I3,3X,1P,10E12.5/(7X,1P,10E12.5))
       DO 7 IM=1,NN
     7 AANS(IM,J)=AANS(IM,J)+ANS(IM,IG,J)
       IG=0
     8 WRITE(NOU,6)IG,(AANS(IM,J),IM=1,NN)
       WA=0.
       WB=0.
       DO 9 IM=1,NN
       WA=WA+AANS(IM,1)
     9 WB=WB+AANS(IM,2)
       AKINF=WA/WB
       WRITE(NOU,10)WA,WB,AKINF
    10 FORMAT('0NUF ',1PE12.5,'  ABS ',E12.5,'  KINF ',E12.5)
C           release storage and set condition code to 1
       CALL VEXIT(1)
       STOP
       END
C
C
       SUBROUTINE VEXIT(N)
C           release memory obtained with VARRAY
```

```
      CALL RARRAY
C           set condition code and exit
      CALL CEXIT(N)
      STOP
      END
C
C
      SUBROUTINE RDGEOM(IU,NN,NMESH,XM,VOL,LAY)
C           read geom data pool
      DIMENSION XM(2),VOL(2),LAY(2),HEAD(18)
      READ(IU)HEAD,ND,IGEOM,NLS
      READ(IU)NMESH,(XM(I),I=1,NMESH)
      READ(IU)
      READ(IU)L,(LAY(I),I=1,L)
      IF(NLS.NE.1)GO TO 6
C           calculate volumes
      X=0.
      V=0.
      DO 5 I=1,NMESH
      X=X+XM(I)
      IF(IGEOM-1)1,2,3
    1 VV=X
      GO TO 4
    2 VV=3.14159*X*X
      GO TO 4
    3 W=4.18879*X*X*X
    4 VOL(I)=VV-V
    5 V= VV
      GO TO 7
C           cluster geometry
    6 READ(IU)
      READ(IU)L,(XM(NMESH+I),I=1,L)
      READ(IU)L,(LAY(NMESH+I),I=1,L)
      READ(IU)NMESH,(VOL(I),I=1,NMESH)
C           NN is material of highest number
    7 NN=0
      DO 8 I=1,NMESH
    8 IF(LAY(I).GT.NN)NN=LAY(I)
      REWIND IU
      RETURN
      END
C
C
      SUBROUTINE RDFLB(IU,NMD,NMESH,NG,FLUX)
C           read FLUXB data pool
      DOUBLE PRECISION FLUX(NMD,2),EIGEN
      READ(IU)
      READ(IU)
      READ(IU)EIGEN,NG,NMESH,((FLUX(I,J),I=1,NMESH),J=1,NG)
      REWIND IU
```

```
      RETURN
      END
C
C
      SUBROUTINE RDXS(IU,NGD,NN,NG,TR,ABS,ANUF,SCAT)
C          read XSLIB data pool
      DIMENSION TR(NGD,2),ABS(NGD,2),ANUF(NGD,2),SCAT(NGD,NGD,2)
      DIMENSION IST(150),ST(150)
      EQUIVALENCE (ST(1),IST(1))
      CALL ARDP(IU,0)
      CALL ARDT(1,1,1,1,1)
      CALL ARDN(DUMY,-20)
      CALL ARDN(IST,10)
      NG=IST(3)
      IF(NN.EQ.0)NN=IST(2)
C          start material loop
      DO 3 L=1,NN
      CALL ARDP(IU,1)
      CALL ARDN(DUMY,-6)
      CALL ARDN(IST,5)
      I=20+NG+IST(1)+IST(2)+IST(3)+IST(4)
      CALL ARDN(DUMY,-I)
C          start group loop
      DO 3 I=1,NG
C          set transport,absorption and nu-fission
      CALL ARDS(LPS,LV,ST)
      TR(I,L)=ST(1)
      ABS(I,L)=ST(2)
      ANUF(I,L)=ST(3)
C          set up full scatter matrix
      DO 1 J=1,NG
    1 SCAT(J,I,L)=0.
      CALL ARDS(LPS,LV,ST)
      K=I-LPS
      DO 2 J=1,LV
    2 SCAT(J+K,I,L)=ST(J)
    3 CONTINUE
      CALL ARDP(IU,0)
      RETURN
      END
```

APPENDIX F
MINOR MODULES OF THE AUS SYSTEM

F1. INTRODUCTION

This appendix describes a number of the minor modules of the AUS scheme. The AUSIDD module is a one-dimensional diffusion module intended for use in multigroup calculations to provide condensation spectra. It is complementary to the multidimensional POW3D diffusion module [Harrington, Pollard & Barry 1996]. GEOM is a simple module which which may be used to enter data onto a *gm*1 data pool. The three simple modules ORNL, MERGEL and JOINER are used for the manipulation of cross sections. The EXPAND module duplicates all the data pool output of a cell calculation for use in initialising some types of global burnup calculation.

F2. AUSIDD — A ONE-DIMENSIONAL DIFFUSION MODULE

F2.1. Description

AUSIDD is a one-dimensional multigroup diffusion module which has been designed to provide neutron flux and adjoint distributions for condensation of group cross sections. Since, the general-purpose diffusion theory module POW3D is not well suited to many-group one-dimensional calculations, AUSIDD has been written to complement POW3D. AUSIDD has the advantage of allowing fission spectra to be material-dependent. It also differs from the POW3D module in having a finite difference scheme with mesh points at the centre of the mesh intervals.

Because it is intended that the use of AUSIDD be restricted to calculations to provide condensation spectra, the module does not include any acceleration of the outer (eigenvalue) iteration. That is, the simple power method is used in eigenvalue problems. In problems with upscatter, an implicit iteration method called CLIMP (Section F2.2) is used to accelerate the convergence of the group fluxes. This iteration over groups is combined with the power iteration. That is, only one pass through the groups is made for each power iteration. In forward calculations, a renormalisation of the total neutron loss to the source is also applied. The lack of acceleration of the power iteration may result in a large number of iterations in eigenvalue problems which have little difference between the two largest eigenvalues.

The module obtains cross sections from an AUS data pool on Fortran unit 10. The module writes an AUS geometry data pool on Fortran unit 7 and an AUS FLUXB data pool on unit 9. A geometry data pool on unit 7 may be used as input.

F2.2. CLIMP Iteration Method

The CLIMP method (Conditional Limited IMPlicit) has been developed to solve the upscattering problem in thermal reactor calculations. It is presented here in terms of a method for accelerating the solution of the simple matrix equation $\sum_j a_{ij} x_j = b_i$. Then the standard Gauss-Seidel method may be represented by

$$x_i^{(n)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(n)} - \sum_{j=i+1}^{N} a_{ij}\, x_j^{(n-1)}}{a_{ii}}$$

Firstly a limited implicit method is introduced, represented by

$$x_i^{(n)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(n)} - \left(\sum_{j=i+1}^{N} a_{ij}\, x_j^{(n-1)} + R\, x_i^{(n-1)} \right)}{a_{ii} - R}$$

where

$$R = \min\left(- \sum_{j=i+1}^{N} a_{ij}\, x_j^{(n-1)} / x_i^{(n-1)} \;,\; f\, a_{ii} \right)$$

and f is a parameter which limits the implicit component to be a fraction of the diagonal term a_{ii}.

In AUSIDD, this process is applied to the scattering and the implicit component is limited to a fraction of the removal cross section from the group at each mesh point.

Tests showed that this process was too sensitive to the parameter f. Hence the following conditions were introduced to form the CLIMP method.

The limited implicit method is used for iteration n unless

- iteration n-1 was implicit and $\varepsilon^{(n-1)} \geq \varepsilon_{LI}$, or
- iteration n-1 was Gauss-Seidel and $\varepsilon^{(n-1)} \geq min(\, \varepsilon_{LI}, \varepsilon_{GS})$, otherwise

Gauss-Seidel is used. Here,

- ε^n is the error after iteration n *i.e.* the maximum fractional difference in point fluxes between the n^{th} and $(n-1)^{th}$ iterations,
- ε_{LI} is the minimum ε^i for previous implicit iterations, and ε_{GS} is the minimum ε^i for previous Gauss-Seidel iterations.

Tests showed that this method was rather insensitive to the value of f and gave improvements of up to a factor of 6 over Gauss-Seidel. Scaling the total neutron loss to the total source gave an improvement in some calculations and is always included in forward calculations. A default value of 0.7 is used for f.

F2.3. Input

The input to AUSIDD is in the form of keywords followed by a string of data items. The specification of geometry is the same as in the MIRANDA module Robinson [1986c] which is compatible with that of the POW3D module. Other data are similar to the ANAUSN module [Clancy 1982]. In the description, the standard conventions (Section 2.) have been used. The data are read with the SCAN free input routine [Bennett & Pollard 1967].

The geometry may be read from an AUS data pool on unit 7 or specified by the following entries:

xm *ibc* $x_1\ x_2 \cdots x_n$ *obc* ,
rm *ibc* $x_1\ x_2 \cdots x_n$ *obc* , or
rm(sphere) *ibc* $x_1\ x_2 \cdots x_n$ *obc*

where slab, cylindrical or spherical geometry respectively are implied,
ibc, and *obc* are the inner and outer boundary conditions, which are either zero for a reflective boundary or the linear extrapolation distance in mean free paths for a free boundary (0.71 is normally used), and
x_i is the width of the i^{th} mesh interval in cm.

reg ($\left\{\begin{matrix}\textbf{mx}\\ \textbf{mr}\end{matrix}\right\}$ $int_1\ int_2 \cdots int_n\ name_1$), ...

where **mx, mr** may be specified or omitted,
int_i are the mesh interval numbers which are filled by the material names and
$name_i$ may have the form of a material name or be **m**(j) which specifies the j^{th} material on the cross-section data pool. Only the first 8-byte word of a material name is given.

Examples

```
rm 0 10*1 10*0.5 0.71
reg 1(1)10 core 11(1)20 blnk
reg mr 1(1)10 m(1) mr 11(1)20 m(2)
```

The other possible entries are

bsq b where b is the buckling in the transverse direction;
eps e_1 where e_1 is the required accuracy in k_{eff}, default 0.0001;
epsp e_2 where e_2 is the required accuracy in group flux at any point, default 0.0005;
uprat f where f limits the implicit component in upscattering problems [default f = 0.7] (Section F2.2);
adjoint is specified if both forward and adjoint calculations are required, the default is forward only;
wrxs causes cross sections to be printed;
wrsf causes the calculated fluxes to be printed;
mqv $int_1\ int_2 \cdots int_n$
gives the set of interval numbers int_i at which a volume source is specified, the default is an eigenvalue calculation;
qv $(s_{11}\ s_{21} \cdots s_{m1})\ (s_{12}\ s_{22} \cdots s_{m2}) \cdots$
where s_{ij} specifies the source density in energy group i for interval int_j, m is the number of groups, and exactly m × nqv values must be given, where nqv is the number of intervals on the **mqv** entry; and
start causes the calculation to begin.

If the geometry data pool is available and the default options are satisfactory, no input data (an empty data set) need be given.

F3. GEOM — A MODULE TO ENTER GEOMETRY DATA

GEOM is an AUS utility module for loading a geometry data pool. It is intended to assist in the use of AUS for editing the flux output of programs such as ANISN and DORT. The input stream is read from Fortran unit 1 and geometry data is written on unit 4. There is no standard linkage for the module, so it is invoked using, for example,

```
link geom(1,2),(4,gm1)
```

The mesh intervals and boundary conditions are given exactly as in POW3D using **xm, ym, zm, rm, rm(sphere)**, as appropriate, where each entry is of the form

xm $d_L\ ,\ \delta_1\ ,\ \delta_2\ ,\ \ldots\ ,\ \delta_n\ ,\ d_R$

where d_L, d_R are left and right boundary conditions and
δ_i is the i^{th} mesh interval of a set of n intervals.

As in POW3D, **reg** entries are used to specify the material layout. The form is

reg **mx** $i_1, i_2, \ldots, i_I$ **my** $j_1, j_2, \ldots, j_J$ **m**(m)
or
reg **mr** $i_1, i_2, \ldots, i_I$ **mz** $j_1, j_2, \ldots, j_J$ **m**(m)
or
reg **mx** $i_1, i_2, \ldots, i_I$ **my** $j_1, j_2, \ldots, j_J$ **mz** $k_1, k_2, \ldots, k_K$ **m**(m)

where the i, j and k are interval numbers, and m specifies the m^{th} material.

F4. ORNL — A MODULE TO GENERATE ORNL CROSS SECTIONS

F4.1. Description

Many of the transport codes distributed by the Radiation Shielding Information Center at the Oak Ridge National Laboratory (ORNL) use input cross sections of similar form. Some of these codes accept cross sections in ANISN-style as part of the input stream and read them with the FIDO input routines. Other codes accept cross sections in the form of an AMPX working library which is in binary form. The ORNL module provides a partial connection between AUS and these transport codes by generating either of these forms of cross sections from an AUS cross-section data pool. All the cross sections required in FIDO format for a transport calculation should be generated in one run of the ORNL module, because the group cross-section table must be of fixed form for all materials. The AUS cross sections are taken from Fortran unit 10. FIDO format cross sections are written on Fortran unit 2 in either free or fixed form. The first FIDO record produced is a comment giving the cross-section table length and the position of the total and self-scatter cross sections. An AMPX working library is written on Fortran unit 11. Normally these Fortran units are connected to the symbolic files *xs1*, *dd12* and *xs2* respectively.

F4.2. Input

The input to ORNL is in the form of keywords, some of which are followed by an integer. In the description, the standard conventions (Section 2) are used. No input data (an empty data set) need be given if the defaults are satisfactory.

nmat *n*

specifies the number of materials to be processed. That is the first *n* materials of the AUS data pool are used. The default is all the materials.

nl *n*

specifies the P_n order *n* of scattering to be used. The default is to use the order on the AUS data pool.

reac *ir*

specifies the AUS reaction number *ir* of an optional additional reaction to be stored in the first position of the FIDO cross section table. The default is no additional reaction.

fixed

specifies fixed format FIDO. The default is free format FIDO.

ampx

specifies generation of an AMPX library rather than FIDO format. The default is FIDO format.

morse

specifies that the cross sections are for use in the MORSE code or any other Monte Carlo code. When the keyword **morse** is used, the total cross section is taken from AUS reaction 5 and the P_0 self-scatter term is adjusted rather than the normal reverse procedure. The simple flux weighting

used with AUS reaction 5 is more suitable for Monte Carlo calculations than the weighting used for the total component in the self-scatter term.

noup

specifies that upscatter terms are to be removed from the scattering matrix by adding these terms into the self-scatter term. This option, which assists convergence in S_N codes, should only be used when upscatter is known to be insignificant or the effect of this approximation has been established.

F5. MERGEL — A MODULE TO MERGE MATERIAL CROSS-SECTION FILES

F5.1. Description

Many applications of the AUS scheme require that a set of cross sections for a global calculation be constructed from a number of cross-section generation calculations. MERGEL is a simple module to merge the contents of two AUS cross-section data pools which have the same group structure. The output is a new data pool containing materials from both the input data pools.

F5.2. Input

The input which is taken from Fortran unit 1 consists of one record used as a title to be written as words 5 to 20 of the output data pool identification.
If the output file is to be direct access, this needs to be followed by a record with the keyword

direct

The set of integers on the following records are read under free format. The required integers are

lib1 lib2 lib3 nn
(lp(i), i=1,nn), (nl(i), i=1,nn)

where *lib*1 and *lib*2 are the Fortran unit numbers of the two input data pools,
*lib*3 is the Fortran unit number of the output data pool,
nn is the number of materials on the output data pool,
$lp(i)$ is the position of the i^{th} output material on the input data pool $nl(i)$, and
$nl(i)$ takes the value 1 for *lib*1 or 2 for *lib*2.

The unit number of *lib*3 may be the same as *lib*1 or *lib*2. In that case Fortran unit 29 is used as a scratch data set. The value of *lib*2 may be 0 if a selection of the materials from *lib*1 is required. If *nn*=0, the output consists of all the *lib*1 materials followed by all the *lib*2 materials.

The information on group structure and other data not related to any particular material are taken from *lib*1. The group energy boundaries on *lib*1 and *lib*2 are tested for compatibility before they are merged. If the values on one of the input data pools are zero, this also is regarded as acceptable provided only that the number of groups is the same on *lib*1 and *lib*2.

An additional option is available for use in cell burnup calculations. If *nn* is negative, the output consists of

$(-nn \times (ncell - 1) + lp(1))$ materials from *lib*1,
$-nn$ materials from *lib*2,
remaining materials from *lib*1, and
remaining materials from *lib*2.

Here *ncell* is the number of cell calculations performed in the burnup run, which is obtained from the *st2* STATUS data pool on Fortran unit 4. The intention is that $lp(1)$ is the number of materials that are independent of burnup, *nn* is the number of macroscopic materials produced in each cell calculation and the remaining materials give microscopic nuclide cross sections.

F6. JOINER — A MODULE TO FORM IRRADIATION-DEPENDENT CROSS SECTIONS

The nuclide cross sections on an AUS cross section data pool for use by the CHAR module [Robinson 1986b] may be irradiation-dependent. The JOINER module is used to construct such a data pool from a normal cross-section data pool without such dependence. The data are combined into an AUS type-4 cross-section data pool in which all cross-section and scattering-matrix data are irradiation-dependent.

The data on Fortran unit 11, which are for one irradiation value, are added to the irradiation-dependent data on unit 10 and the output written on unit 12. No card image input is required. The nuclides on unit 11 must be in the same order as the nuclides on unit 10. Unit 10 may contain more nuclides than unit 11 and these nuclides are written unchanged on unit 12.

F7. EXPAND — A MODULE TO DUPLICATE OUTPUT FROM A CELL CALCULATION

F7.1. Description

The AUS system may be used to perform global burnup calculations in which at each burnup step a cell calculation is performed for each reactor zone to provide material cross sections and microscopic nuclide cross sections for that zone. The EXPAND module, which duplicates the data pool output from a cell calculation, is useful for initiating such a calculation.

The data pools input to the module are the pair of STATUS data pools *st1* and *st2* on Fortran units 4 and 5, respectively, macroscopic material cross sections on Fortran unit 10 and, optionally, microscopic nuclide cross sections on unit 11. A scratch data set is required on Fortran unit 26. The output consists of updated data pools in which all output for nominated cells has been duplicated and put in a requested order. Cell names must be exactly four characters because the module generates six-character cell names, leaving the last two characters of the eight-character cross-section data source word to describe the region within a cell. If nuclide cross sections are not updated, neither are the nuclide (as distinct from material) names on the STATUS data pool.

The module provides the facility to condense the spatial smearing factors within a cell in calculations in which the smearing of cell cross sections is followed by a further condensation step. A condensation of the smearing factors is required if the nuclide cross sections are also condensed in the second step. The EXPAND module is the only AUS module which provides for condensation of smearing factors.

F7.2. Input

The input on Fortran unit 1 consists of a string of numeric and alphanumeric data which are read by the SCAN [Bennett & Pollard 1967] free input routine. The data are

ntype, *ifxs2*, *jst*, *jcond*, *dt*
(*cellna*(i), *ncell*(i), (*ipos*(j, i), $j = 1, ncell(i)$), (*lab*(j, i), $j = 1, ncell(i)$), $i = 1, ntype$)

where
- *ntype* is the number of cells to be duplicated or repositioned,
- *ifxs2* is 1 if microscopic nuclide data are to be updated, else 0,
- *jst* is 1 if the *st1* data pool is to be added to, or 0 if *st1* is to be restarted,
- *jcond* is the number of energy groups into which spatial smearing factors are to be condensed, or zero if no condensation,
- *dt* is the time step in days, given to assist identification only,
- *cellna* is the cell name of an input cell calculation to be duplicated or repositioned,
- *ncell* is the number of output cells of this type,
- *lpos* is a set of integers giving the position of cells in the output data pools, and

lab is a set of labels used to modify the input cell names, which may be integers less than 100 or words consisting of two alphanumeric non-blank characters.

Example

For input data pools giving data for three cells in the order `refl`, `core`, `blnk` to obtain output for `refl`, four copies of `core` and three copies of `blnk`, the required input is

```
2 0 0 0 0.001
core 4 2345 1234
blnk 3 678 123
```

The output data pools would contain data for the cells named
`refl`, `core01`, `core02`, `core03`, `core04`, `blnk01`, `blnk02`, `blnk03` in that order.

APPENDIX G
INTERACTIVE PLOTTING MODULES OF THE AUS SYSTEM

G1. INTRODUCTION

This appendix describes two interactive plotting modules of the AUS system which are invoked directly from the command line rather than being linked from within AUS under the control of AUSYS. The PLOTXS module may be used to plot neutron and photon cross sections from any AUS XSLIB data pool, including the main cross section libraries. The AUSPLOT module may be used to plot fluxes and reaction rates following a 1D, 2D or 3D neutron diffusion or transport calculation in which the XSLIB and geometry data pools and an appropriate flux file have been saved.

G2. PLOTXS — An Interactive Module For Plotting Cross Sections

The PLOTXS module is an interactive module with a simple menu interface which may be used to plot the cross sections from one or more XSLIB data pools. PLOTXS is invoked using the command:

plotxs [*filenam*]

where *filenam* is the optional name of the file from which cross sections are to be taken. The default file is the main library *aus/endfb6*.

Either neutron or photon group cross sections may be plotted as a function of energy. Plots and optional print may be directed either to the screen or to a file for producing hardcopy later. The XYPLOT package from the ANSTO subroutine library is used to produce the plots. The data for different nuclides from several files may be compared. The module prompts for the required actions. The user may select

- particle (neutron or photon),
- nuclide,
- temperature,
- cross sections identified by AUS numbers or ENDFB MT numbers for additional reactions,
- the plot type and filename, or
- the print format and filename, and
- the plot scale by setting the parameters for the XYPLOT routine XYPAPE.

Some of the options are:

Change the input file
Read new data from the same file
Plot the same data again
Keep the current data for comparison with next cross section
Ratio – divide current data by first cross section and plot
Print the last data – *i.e.* numerical output
Quit

G3. AUSPLOT — An Interactive Module For Plotting Fluxes and Reaction Rates

The AUSPLOT module provides a graphical user interface (GUI) for plotting fluxes, cross sections and reaction rates which have been generated within AUS. The cross sections and geometry data pools used in the flux calculation need to have been saved along with the flux output in the form of a (2D or 3D) FLUXA data pool, a FLUXB data pool or a DORT VARFLM file. A series of options allow a specified quantity to be plotted as a function of one or two independent variables.

AUSPLOT is invoked using the command

ausplot [*xsfile* [*fluxfile* [*geomfile*]]]

where *xsfile*, *fluxfile*, and *geomfile* are the names of the cross section, flux and geometry files respectively. AUSPLOT contains an internal file browser which can be used to select the three required files. If all three files are specified they will be loaded automatically. Otherwise, the remaining files must be selected using the file browser.

AUSPLOT does not do the plotting directly. All plotting functions are implemented using *gnui* which is a GUI for the public domain *gnuplot* plotting program. When AUSPLOT is launched it starts *gnui* as a background process if it is not already running. Control over options such as labels, line styles and axis limits is obtained using the *gnui* window. When edge mesh fluxes are given (FLUXA data pool) they are first averaged to form mesh average data. The spatial data and the energy group data are treated as being point data at the centre of the spatial mesh and at the average lethargy of the energy group in performing the plots. When flux is plotted against energy, the flux per unit lethargy rather than group flux is used. The spatial variation of the flux may be plotted for supergroups, which are selected using the **Groups** button, or for a selected energy. The total flux within the supergroup or the flux per unit energy at the selected energy is displayed.

The main GUI buttons in AUSPLOT are:

Read displays the file browser window used to select the input files. Other AUSPLOT windows are frozen while this window is current.

Groups displays a window which allows editing of the supergroup boundaries. By default these define thermal and epithermal groups with a boundary at 0.625 eV.

Plot plots the currently defined function. This button becomes available only when
a) the data files have been loaded, and
b) the number of independent variables is one or two.

Options displays a window of the four options:
1) Plot single or multiple plots,
2) Use energy or lethargy as energy axis,
3) Select window font size from 10, 12 or 14 point,
4) Choose horizontal or vertical format for main window;
and a divide button which is to be pushed between specifying one function which is the numerator and another which is the denominator.

Quit displays a dialogue box asking for confirmation before exiting AUSPLOT.

MIRANDA – A MODULE BASED ON MULTIREGION RESONANCE THEORY FOR GENERATING CROSS SECTIONS

1. INTRODUCTION

The AUS code system was originally developed at Lucas Heights for reactor neutronics computations of a wide range of thermal and fast fission reactor systems. AUS was extended for fusion blanket calculations in the mid 1980s and it then became also applicable to many other neutronics applications. MIRANDA is the cross-section generation module for AUS and it is suitable for all the diverse calculations to which AUS is applied. The term 'cross-section generation' is used in the restricted sense of preparing group cross-section data, suitable for a particular subsystem of a reactor or an assembly of material, from an input group cross-section library of wide application.

MIRANDA uses as input data a multigroup cross-section library with group scattering matrices of any P_n order, resonance cross sections in the form of subgroup data, and scattering matrices dependent on potential scattering. All relevant quantities may be temperature dependent. The first two standard libraries for MIRANDA contained data derived mainly from ENDF/B-IV. The original 128-group library *aus/endfb* was developed for fission reactor core calculations and contains neutron data only. The next library *aus/endf200g* which had 200 neutron groups and 37 photon groups was intended for all applications. The fission product data on this library were derived from ENDF/B-V. The current library *aus/endfb6* has the same group structures as *aus/endf200g* but was derived from ENDF/B-VI. The use of these libraries with groups as wide as 0.25 lethargy requires that most major nuclides be given a resonance treatment.

The basic output from MIRANDA is a cross-section data pool containing cross sections for each region of a lattice cell for use in a transport calculation of the cell. Thus, MIRANDA is only the first phase of a normal reactor lattice calculation. The major function of MIRANDA is the resonance theory calculation of nuclide cross sections, in the library group structure, appropriate to the system under study. The multiregion resonance calculation may be performed in slab, cylinder or cluster geometry using the subgroup method. Preliminary group condensation may also be carried out following a homogeneous B_L flux solution. In applications with no cell structure, MIRANDA directly provides cross sections for a global calculation of the system.

The first MIRANDA report [Robinson 1977] should be referred to for details of the methods of calculation; only the modifications to those methods are detailed here. In addition, this report is intended as a user's manual. It contains an outline of the code which should be sufficient for most users, a complete description of the input data and the current contents of the *aus/endfb6* library. This report replaces the second MIRANDA report [Robinson 1985]. However, to obtain the content and method of generation of the earlier cross section libraries, reference to Robinson [1985] should still be made.

A comparison of MIRANDA resonance calculations with numerical solutions using the PEARLS code [Chiarella 1971] was given by Robinson [1977], and a more recent comparison with the MCRP Monte Carlo code [Doherty and Robinson 1985] for thermal reactor lattices has been given by Robinson [1985]. These comparisons were performed using the point cross-section files from which the parameters used in the MIRANDA resonance treatment were derived. The general conclusion was that the accuracy in resonance captures is usually about 1% but this may deteriorate when there are large resonance overlap effects between resonances in different nuclides.

2. OVERVIEW OF THE CODE AND LIBRARY

2.1. Cross-section Library

The cross-section library is in the form of an AUS XSLIB data pool, the structure of which has been given by Robinson [1987]. The current library *aus/endfb6* was generated from ENDF/B-VI as described in some detail by Robinson [1993]. In essence much of this was performed using NJOY [MacFarlane et al 1982] but a number of programs specific to AUS requirements were used as well, particularly for the resonance treatment.

The group structure of the libraries and the nuclides included are detailed in Tables 1 to 4. Each nuclide has a three part name. The first part gives the simple name and the remainder describes the origins of the nuclide. The nuclide labelled *zz999* is a pure absorber with inverse velocity energy dependence and a cross section of 1 barn at 0.025 eV. The nuclides named *h0zrh* and *zr0zrh* are H and Zr in Zr-H. Some nuclides are identical in ENDF/B-V (part of which was included on *aus/endf200g*) and ENDF/B-VI. This is indicated by the second part of the full nuclide name being *endfb5*. The third part of the name for ENDF/B-V nuclides is used to indicate that the AUS cross sections were not reprocessed (*orig*) or were reprocessed by using NJOY (*njoy*). For ENDF/B-VI nuclides, the third part of the name indicates the ENDF/B-VI revision number and any AUS revision number. The ENDF/B-VI files correspond to revision 1 of the whole file and the revised nuclides are thus denoted *m010* for simple isotopes or *m0n0* for elements in which n constituents have been revised. The last character is reserved for AUS revisions.

The representation of any nuclide on the library is at least as detailed as on the previous AUS library in terms of representation of P_l expansion, temperature effects, resonance shielding and photon data. The P_l order given in the table is for fast neutron scattering. Thermal neutron scattering and photon production have a P_3 expansion at most. The photon interaction has a P_5 expansion for all data given. The nuclides with resonance treatment are indicated in the table; the remainder are for infinite dilution. The table indicates those nuclides without photon data. Such nuclides also have no kerma data. Users should take care that the data on the library are appropriate for the intended use. In particular, fission products have neutron cross sections only, no energy deposition and no thermal scattering. Their scattering expansion is P_1.

The library includes additional cross sections for many of the lighter nuclides. The previous libraries just included (n,γ) and (n,2n), except for some nuclides where another reaction was given instead. Now there is a set of reactions which should satisfy all requirements (except for individual reactions of an isotope included in an element). These reactions include (n,p), (n,α), gas production and neutron damage. On the *pion* computer at ANSTO, further information on a nuclide may be obtained using the *plotxs* command, which enables any cross sections on the library to be plotted interactively. The command also lists the set of additional reactions and any comments on a requested nuclide. The reactions are identified by ENDF MT numbers. If further detail on the generation of any nuclide *xyz* is required, the input files used may be accessed on the *pion* as files *xyz.** in the directory */home/gsr/work/xs6*. For further information on the neutron cross section data, the original ENDF/B-VI files may be accessed as individual files for each material in the directory */home/gsr/xs/endfb6/neutron*.

Resonance data in the library are given mainly in the form of subgroup parameters. These data have been obtained from group resonance integrals per unit lethargy I, tabulated as a function of effective potential scattering cross sections, by fits of the form

$$I(\hat{\sigma}_p) = \sum_i w_i \, \hat{\sigma}_p \, / \, (\, \sigma_i + \hat{\sigma}_p \,) \, ,$$

where σ_i is a parameter representing the total cross section of the nuclide in the i^{th} subgroup, and

w_i is the subgroup weight. Resonance integrals have been obtained by generating cross sections on a fine energy mesh for use in numerical solutions of the neutron slowing-down equation for mixtures of the resonance nuclide and hydrogen.

The calculation also provides the P_0 scattering matrix as a function of σ_p. This treatment has also been applied in the unresolved region, using a ladder of resonances obtained by selecting parameters from the distributions about the average resonance parameters. Further detail of the methods for generating data on *aus/endfb6* were given by Robinson [1993]. For those fission product nuclides which were carried over from *aus/endf200g* to *endfb6* because the ENDF/B-VI data was identical to that on ENDF/B V, a modified version [Robinson 1981] of the XLACS2 code was used rather than NJOY.

2.2. Resonance Treatment

The resonance treatment in MIRANDA is a multiregion calculation which makes use of a set of collision probability routines to obtain subgroup fluxes for each resonance nuclide in turn, then calculates the resonance integrals. Scattering is assumed to be elastic and spherically symmetric in the centre-of-mass system. The collision probability routines included are the numerical integration routines for reflected slab, translational slab and reflected cylinder developed by Doherty [1969], and the approximate routines for cylinders and clusters developed by Robinson [1979]. The approximate routine is normally used for cylinders as it is much faster, gives adequate accuracy, and provides a treatment of polygonal boundaries. Evaluation of the resonance integral does not depend explicitly on an equivalence relation but a numerical equivalence relation is derived to obtain the outscatters of the resonance nuclide.

Corrections to the simple treatment outlined above include the following:
- a λ method to form an equivalence between scattering by different nuclides [see Robinson 1977];
- a non-isolated resonance correction;
- a resonance overlap correction; and
- a dependence of group removals on the group resonance escape probability.

The group reaction rates derived from the resonance integrals are preserved when obtaining effective group cross sections. This involves an iteration on the collision probability formulation of the group equations with an asymptotic source to give a set of cross sections and consistent region fluxes which reproduce the required reaction rate.

Details of the resonance treatment were given by Robinson [1977]. Since then, the treatment has been slightly modified. The changes are discussed in Section 4.

2.3. Homogeneous Flux Calculation

A homogeneous model of the lattice is obtained by spatial averaging, using volume weighting in nonresonance groups and the fluxes obtained for an asymptotic group source in resonance groups. The equations solved are either the transport corrected P_0 equations or the B_L equations for $L > 0$. A search for critical buckling may be performed. The photon flux may be calculated using the photon production from the calculated neutron flux as the source. If bilinear weighting is requested, the adjoint flux equations are also solved. The method of solution has been described by Robinson [1977].

2.4. Condensed Cross-section Output

The flux used in energy condensation is formed by combining the homogeneous group flux with the spatial fluxes available in resonance groups. Only limited condensation should be performed in MIRANDA, because in most cases this flux is very approximate. For flux weighting, the

condensation procedures are straight forward except for the transport cross section for which the formulae adopted are based on the use of the derived cross sections in global calculations. The procedures have been detailed by Robinson [1977]. For bilinear weighting, the condensation procedures are the same as those of the EDITAR module [Robinson 1986].

3. USE OF AUS DATA POOLS

MIRANDA normally outputs the following data pools:

- A cross-section data pool on FORTRAN unit 10 containing macroscopic cross sections for the materials of the lattice, and any desired microscopic cross sections.
- A geometry data pool on FORTRAN unit 7.
- A FLUXB data pool on FORTRAN unit 9 giving a P_0 flux in the spatial mesh of the geometry data pool, for the condensed group structure, which serves as a flux guess for the next module.
- Additional entries added to the pair of STATUS data pools, ST1 on FORTRAN unit 4 and ST2 on FORTRAN unit 25.

The purpose of the entries on the STATUS data pool has been outlined in Robinson [1975]. The following actions are taken by MIRANDA.

If the ST2 data set is empty, both ST1 and ST2 are initiated with TIME entries for time zero. The module then enters the write mode in which data come entirely from the input stream and entries are added to ST1 and ST2. A CELL entry is added to ST2 and CELL, GRPS and material definition entries are added to ST1. A material definition is written for each output material which is not a nuclide on the input cross-section library.

If ST2 is not empty, the last CELL entry acts as a counter of calculations already performed. If there are no more CELL entries for the current time on ST1 than on ST2, the module enters the write mode as above. If there are further CELL entries, the first entry sets the cell name and the following entries are used for material definitions which takes precedence over the input stream. When data are obtained from ST1 in this way (which should only occur in burnup calculations), the only STATUS output is a CELL entry on ST2.

4. MODIFICATIONS TO THE RESONANCE TREATMENT

The chief modification is simply the replacement of the collision probability routines for cylinders and rod clusters. The results for cylinders with a white reflective circular boundary are practically unchanged but the new routines provide an approximate treatment of polygonal boundaries. The new cluster routines are a considerable improvement on those used originally.

The inclusion of photon production and kerma factor data in the calculation was easily accommodated. For resonance nuclides, the photon production data stored in the library consists of separate multiplicity matrices for capture, fission and a composite of other reactions. Thus photon production from capture and fission are resonance-shielded in the same way as the neutron reaction data. Similarly, separate kerma factors are stored for capture, fission, elastic scattering and a composite reaction, and thus can be shielded appropriately.

The original MIRANDA resonance treatment relied on resonance nuclides scattering neutrons down through only one energy group. The following changes have been made to generalise that treatment; for continuity, the notation and equation numbers are as used in the earlier report [Robinson 1977]:

$$\phi_0 = 1 / \sum_i V_i \left\{ \sum_m N_{mi} \xi_m \sigma_{pm} + \sum_{m \in L_l} N_{mi} \left(\tau \sigma_m(\bar{S}_{mi}) \, \xi_m / \beta_m - \xi_m \sigma_{pm} \right) \right\} \qquad (5.33a)$$

$$\phi'_0 = 1 / \sum_i V_i \left\{ \sum_{m \neq l} N_{mi}\sigma_{pm} + N_{li}\xi_l\sigma_{pl} + \sum_{m \in L_l} N_{mi} (\tau\sigma_{rm}(\bar{S}_{mi}) / \beta_m - \sigma_{pm}) \right\} \quad (5.34a)$$

$$\gamma_l = T_0 / \left\{ T_0 + \sum_i V_i N_{li} (\beta_l\sigma_{pl} - \tau\ \sigma_{rl}(\bar{S}_{li})) \right\} \quad (5.76)$$

where

$$T_0 = \sum_i V_i \left\{ \sum_{m \notin L} N_{mi}\sigma_{pm}\beta_m + \sum_{m \in L} N_{mi}\ \tau\sigma_{rm}(\bar{S}_{mi}) \right\}$$

The remaining changes relate to the provision of an optional method which may be more suitable than the standard treatment in heavily absorbing systems such as fast reactor assemblies below neutron energies of 1 keV. An option is available to treat some nuclides by a pseudo narrow resonance (NR) treatment. In this option, the Hill-Schaeffer λ parameter is applied only to the resonance nuclide and the ρ correction factor (equation 5.32 of Robinson [1977]) is not applied. This is as close as possible to an NR treatment, with subgroup parameters fitted to resonance integrals treated as a function of σ_p. The results for mixtures of the resonance nuclide and hydrogen are the same under this option and the standard treatment.

A further option provides a very approximate method of allowing for the effect on resonance overlap of the position of resonances within a neutron group. This effect is based on the group resonance escape probability p, and not on the overlap of resonances at a particular energy. Factors giving the average position within the group of resonance absorption and resonance fission have been included on *aus/endf200g* and *aus/endfb6* for some actinides. Resonance integral weighted values of $log(E_r/E_g)/log(E_{g+1}/E_g)$ have been formed using the same resonance integrals as were used to form average λ parameters.

The correction factor applied in MIRANDA to the absorption integral of nuclide k is given by

$$exp(\sum_m N_m\ I_m\ (f_m - f_k)\ \phi\ )\ ,$$

and the correction factor for the fission integral is

$$exp(\sum_m N_m\ I_m\ (f_m - f_k^f)\ \phi\ )\ ,$$

where f_m and f_m^f are position factors of nuclide m for absorption and fission respectively. Where not given, the position factors are taken as 0.5.

5. INPUT DATA

5.1. General Layout

Input data to MIRANDA are in free format, with keywords to indicate the data type followed by a string of numeric or alphabetic information. The data are entered in columns 1 to 72 with each keyword starting on a new record. As the data are read with the SCAN input routine [Bennett and Pollard 1967], all the facilities and conventions of SCAN apply. The readability of the data may be improved by including any desired special characters.

The following conventions have been used in this report:

- information to be reproduced exactly is given in **bold**,
- items that the user will replace with an actual value are given in *italics*,
- omission of some data is indicated by ...,
- examples and defaults are given in `constant width` type,
- optional items are given inside [ ],
- items from which the user will choose are listed in a column inside { }.

Many of the input variables have been given default values. Thus, if the standard value is required, an entire keyword and data may be omitted or trailing data may be left off the end of a data string. There is no set order in which the keywords must be given, but some of the data require information to be previously defined. For example, a material may not be referred to before it is defined. It is, therefore, best to follow the order of keywords given below. When a series of calculations is undertaken, only the data which are to be altered need be given for subsequent calculations.

In the description of entries given below, "*" indicates those that should suffice for most calculations.

5.2. Heading Information — head *

head *Jobname Heading*

where *Jobname* is a word of 1 to 6 alphanumeric characters which provides a unique label for all output including each material on an output cross-section data pool, and
Heading is a string of characters, including blanks, also used for labelling.

Example:

```
head sr5 test calculation
```

5.3. Problem Size Specification — prelude

prelude **mdef** = *md* **mreqd** = *mr* **maxx** = *mx* **maxbn** = *mb*

with default values

```
prelude mdef = 50 mreqd = 20 maxx = 20 maxbn = 0
```

where *md* is the maximum number of defined materials,
mr is the maximum number of output materials,
i.e. the number of materials on the **reqd** entry,
mx is the maximum number of geometry intervals, and
mb is the maximum order of a B_L flux solution.

The **prelude** values of **mreqd**, **maxx** and **maxbn** may be exceeded in the first calculation, which serves to redefine these maxima, so that a **prelude** entry is seldom required. If, however, a subsequent calculation requires a larger value than the first, a **prelude** entry is necessary.

Example:

```
prelude maxbn=3 mdef=60
```

is required for a B3 calculation following a B1 calculation with 60 materials defined.

5.4. Library Selection — use

use *lib last*

with default

```
use  8  pfp
```

where *lib* is the FORTRAN unit number of the input cross-section library, and
last is the name of the last nuclide on the library to be included.

The specification of *last* has application only in burnup calculations to alter the set of fission products which are included. The AUS libraries have a set of individual fission products and a pseudo fission product `pfp`, followed by a further set of fission products of low reactivity worth which are not usually included. In normal burnup calculations, MIRANDA uses the decay and yield data of the discarded fission products to modify the yield data of those kept. The `pfp` fission product is used only if it is the last nuclide included. Thus

```
use 8 all
```

may be used to include all the individual fission products and discard `pfp`.

A **use** entry serves to initialise variables to the standard state, causes a preliminary reading of the cross-section library and causes the STATUS data pool to be read. If a **use** entry is not given before any input entry except **head**, **prelude** (or **test**) is given, the code generates the standard **use** entry.

5.5. Library Selection — select

select *name*, *source*, *mod* = *newname*

where
- *name* is the name of a library nuclide for which a non-standard set of data is required,
- *source* is the data source of the required data,
- *mod* if given, selects the appropriate data modification, and
- *newname* if given, renames the nuclide so that two versions of a nuclide may be included in a calculation.

The nuclides on the library are labelled by 20 characters consisting of an 8-character simple name, an 8-character data source and a 4-character modifier. The library may include more than one version of a nuclide, and the one having the highest update number is normally used. A number of **select** entries may be used to choose non-standard versions of nuclide data, but they must immediately follow a **use** entry or be the first entries apart from **head**, **prelude** or **test**. The nuclide list for *aus/endfb6* is given in Tables 3 and 4.

Example (using *aus/endf200g*):

```
select al endfb4 orig = ala
```

selects an old set of aluminium data and renames it `ala` so that it may be compared with the standard aluminium data.

5.6. Default Temperature — temp *

temp *t*

with default

```
temp  300
```

where *t* is the default temperature (K) for any following material definitions which do not include a specific temperature value.

5.7. Material Definition — defn and solution *

The **defn** entry is used to define

- microscopic materials by atom fraction or percent by weight of nuclides or other microscopic materials,
- macroscopic materials by volume fraction of other macroscopic materials, or
- macroscopic materials in terms of atom densities or densities in g cm^{-3}.

Most atomic masses are taken from the cross section library but some data on light elements and molecules is built into the MIRANDA input routine.

defn { **atom** / **gram** / **wt%** } *matname* [*t*] $name_1$, $conc_1$, $name_2$, $conc_2$, ...

where
- *matname* is the 1- to 8-character name to be given to the defined material,
- *t* is the temperature (K) of the material, which may be omitted,
- $name_i$ is the name of a library nuclide or a previously defined material, and
- $conc_i$ is the quantity of $name_i$ to be included, which for **atom** is given in 10^{24} atom cm^{-3} or in atom (or volume) fractions. For **gram**, the quantity is in g cm^{-3}, and for **wt%** it is in weight percent.

The default is **atom** if the type of information is not specified. Values given as weight percent are not renormalised to 100 percent. The following example illustrates most usages. It should be noted that, if u were redefined in a subsequent case, any definitions of materials in terms of u would have to be repeated.

```
defn wt% u u235 3.3 u238 96.7
defn atom uo2  u 1.  o  2.
defn fuela 450. uo2 .022
defn fuelb 600. uo2 .022 b10 1.e-5
defn gram clad al 2.7
```

The set of nuclides currently included in the libraries is given in Appendix A. The names they have been given are the obvious isotope or element name.

The **solution** entry is more specialised than **defn** and is used to define macroscopic materials in which solvent densities are computed by MIRANDA.

solution *matname* [*t*] **density** ρ_s
[$matd1_1$ $matd2_1$, $rhod_1$ $molv_1$; $matd1_2$ $matd2_2$, $rhod_2$ $molv_2$; ...]
with $mat1_1$ $mat2_1$, ρ_1; $mat1_2$ $mat2_2$, ρ_2 ; ... ; **solvent** *mats*

where
- *matname* is the 1- to 8-character name to be given to the defined material,
- *t* is the temperature (K) of the material, which may be omitted,
- ρ_s is the density of the solution, or the density of a solvent used in calculating the density,
- $matd1_i$ is the name of a solute used to calculate the solution density,
- $matd2_i$ is the name of a constituent of $matd1_i$,
- $rhod_i$ is the density of $matd2_i$ in the solution,
- $molv_i$ is the volume (cm^3) of one mole of $matd1_i$,
- $mat1_i$ is the name of a solute used to define *matname*,
- $mat2_i$ is the name of a constituent of $mat1_i$,
- ρ_i is the density of $mat2_i$ in the solution, and
- *mats* is the name of the solvent whose density is determined.

All material names used in the definition of *matname* may be nuclides or defined microscopic materials. There must be only one part of $matd2_i$ in $matd1_i$ and similarly for $mat2_i$ in $mat1_i$. For example, defining B2O3 content in terms of B density would give an incorrect result. All densities are given in g cm^{-3}. The density of the solution may be specified directly or calculated by MIRANDA from the given solvent density. The extra data following **density** is used to determine the volume fraction and density of the named solutes $matd1_i$ and hence derive the solution density from the solvent density. The solutes and densities following **with** are used to specify the

composition of the solution. The density of the named solvent (usually H2O) is determined from the solution density. In general, the solvent and solutes used in determining the solution density are different from those used to specify the composition of the solution.

The following example gives the input for a uranyl-nitrate solution in which the densities of uranium and the nitrate ion were 20.12 and 19.24 grams per litre respectively. The density of the solution is obtained by considering it to be a solution of UO3 in nitric acid (Clark 1982). The relevant acid density and the molar volume of UO3 have to be specified by the user. The composition is then specified in terms of UO3 and HNO3 dissolved in H2O. For this example, MIRANDA would calculate a solution density of 1.0307 which it then uses to determine the density of H2O.

```
defn no3 n 1. o 3.
defn wt% u u234 1.04 u235 93.18 u236 0.27 u238 5.51
defn uo3 u 1. o 3.
defn hno3 h2o 0.5 n 1. o 2.5
solution mix density 1.00877 uo3 u 0.02012 26.528
   with uo3 u 0.02012 hno3 no3 0.01924 solvent h2o
```

5.8. Output Material Selection — reqd *

reqd $name_1$ [(r_1)], $name_2$ [(r_2)], ...

where $name_j$ is the name of a library nuclide or a defined material which is required for output, and

r_j if given, selects the resonance geometry interval number for which the nuclide or material is to be produced.

An interval number r_j is not normally given as the code averages over the required region. The following is a description of the default values. For a defined material, average cross sections are formed over the cell regions occupied by that material (both for materials explicitly in the system, and for intermediate materials such as uo2 above) or over the entire cell for materials not in the cell. For a library nuclide in the cell, a choice may be made between one set of average cross sections or a set of cross sections for each resonance region, which includes the nuclide and is filled by a different material. The choice is made using the **reqdreg** entry, with the default being the preparation of a number of cross section sets for fuel nuclides only.

Output may be requested for the special material `cell`, which has the average properties of the lattice cell. If output of a resonance nuclide is requested for a resonance region in which the nuclide concentration is zero, infinite dilution cross sections will be used. Correct cross sections of a resonance nuclide in such regions may be generated by including a small amount, say 1.E-20, of the nuclide in the material definitions. This procedure is useful in preparing cross sections for a reaction rate scan across the lattice.

If the library order is known, a group of nuclides may be selected using

reqd $name_1$ **to** $name_2$, ...

Examples:

```
reqd  fuel, mod,  u235 to pu241
reqd  fuel, can, mod, refl(3)
```

5.9. Geometry Size Specification — xm or rm *

xm *ibc* x_1, x_2,..., x_n *obc*
rm *ibc* x_1, x_2,..., x_n *obc*

where **xm** implies slab geometry,
rm implies cylindrical geometry,
ibc is normally the inner boundary condition,
x_i is the width of the i^{th} mesh interval in cm, and
obc is the outer boundary condition.

In slab geometry, the boundary conditions have the values 0 for reflective and -1 for translational conditions. In cylindrical geometry, the inner boundary condition is always reflective and additional meanings are given to *ibc* and *obc*. The simplest use is *ibc* = *obc* = 0 for a white reflective boundary condition on a circular outer boundary. The other use is a reflective boundary condition on a polygonal outer boundary which is indicated by a negative value of *obc*. The polygon has *ibc* sides and the |*obc*| innermost intervals are treated as fuel. (This boundary condition is limited to a two-region system. The actual geometry is volume smeared into two regions in applying the boundary condition.) The outermost radius, $\sum_n x_n$, is the radius of a circle having the same area as the polygon.

Either an **xm** or **rm** entry is always required, and homogeneous cases should be run as a one-region calculation.

Example:

```
rm 0 0.5 0.25 0
```

specifies a two-region cylinder with a rod radius of 0.5 cm and an outer radius of 0.75 cm.

5.10. Geometry Layout Specification — reg *

reg [**mx**] (int_1 , ..., int_n , $name_1$) , ...

or

reg [**mr**] (int_1 , ..., int_n , $name_1$) , ...

where **mx** or **mr** may be included for compatibility with other AUS modules or omitted,
int_1, ..., int_n are the mesh interval numbers which are filled by the defined material, $name_1$, and
$name_1$ may have the form of a material name, or be **m**(*j*) which specifies the j^{th} material on the previously given **reqd** entry.

Examples:

```
reg  1  fuel  2  mod
reg  mx 1 3 5 7 m(1)  mx 2(2)8  m(2)
```

5.11. Cluster Specification — rodsub and array *

In cluster geometry, the composition and position of fuel rods within the annular geometry given by the **rm** and **reg** entries are specified by sets of **rodsub** and **array** entries. The material specified by the **reg** entry fills that part of the annulus not occupied by the rods. The boundary conditions specified on the **rm** entry are applied to the inter-rod boundaries when the synthetic collision probability routine is used. The outer boundary of a cluster is always taken to be a circular white reflecting boundary. The cluster geometry is currently limited to equally spaced rods whose centres lie on a circle. The entries have the form

rodsub j $(dr_1, \cdots, dr_n)$, $(name_1, \cdots, name_n)$

where j is the number of the ring, numbered from the centre outward,
n is the number of radial subdivisions of a rod in this ring,
dr_i is the width in cm of the i^{th} radial subdivision, and
$name_i$ is the name of the material in the i^{th} subdivision.

array j, *nrod*, *prod*, *qrod*

where j is as defined above,
nrod is the number of rods in the ring,
prod is the radius of the circle on which the rod centres lie, and
qrod is the angular displacement of one rod from a reference diameter of the cluster (in radians or degrees). The angular displacement is not used in MIRANDA but is entered in the AUS geometry data pool.

Example:

```
defn fuela u238 .048 u235 .0048
defn fuelb fuela 1.
defn mod d2o .033
reqd fuela fuelb mod u235 u238
rm 6 0.985 1.62 5.858 -1
reg 1 2 3 mod
rodsub 1 0.825 fuela
rodsub 2 0.825 fuelb
array 1 1 0
array 2 6 1.875
```

This specifies a 7-rod cluster of unclad fuel pins with a hexagonal boundary applied between pin-cells. Different fuel names are used for the central rod and the rods of the ring, in order that two sets of cross sections will be prepared. Two different sets of ^{235}U and ^{238}U data are also prepared. If one fuel name had been used, average cross sections for the fuel, ^{235}U and ^{238}U would be produced.

5.12. Resonance Geometry Specification — resreg *

The **resreg** entry specifies the geometry to be used in the multiregion resonance calculation and is usually required. A blank **resreg** entry indicates that the geometry is that given by the **xm** or **rm** and **reg** entries. However, there is little point in including a large number of mesh intervals in the resonance calculation. Not only can this become rather expensive in computer time, but the results obtained with a single mesh interval in each physical region should be equally reliable in most cases. The collision probability approximation of a constant source within a mesh interval is completely satisfied for the off-resonance source term of the resonance calculation. Additionally, the Bonalumi approximation normally used in cylindrical geometry becomes worse as the fuel pin is subdivided. A collision probability routine using numerical integration is available for calculations of resonance reaction rate distributions within a fuel pin. A comparison of MIRANDA with Monte Carlo calculations [Robinson 1985] resulted in a recommendation that one mesh interval per physical region should be used in single-rod lattices.

A **resreg** entry is used to smear the geometry given by the **xm** or **rm** and **reg** entry. No provision has been made for smearing the rod subdivisions of cluster geometry. The form of the entry is

resreg $ibc,\ x_1, \ldots, x_m\ ,\ obc$ **smear** $i_1, \cdots, i_n$

where

$ibc, x_1, \cdots, x_m, obc$	specify an m region geometry to be used in the resonance calculations, with the variables having the same meaning as given on the **xm** or **rm** entry,
$i_1, \cdots, i_n$	are the resonance region intervals into which each main geometry interval is to be smeared, and
n	is the number of main geometry intervals.

That is the i_j are integers with values between 1 and m.

Example:

```
xm  -1  4*0.5  -1
reg  1 fuela  3 fuelb  2 4 mod
resreg  0  2*0.5  0  smear  1 2 1 2
```

This specifies a 4-region transitional slab with 2 different fuel regions, which is treated in the resonance calculation as a 2-region reflected slab in which the fuelled regions are smeared together.

5.13. Group Buckling — buck *

buck [bn] $b_1^2,\ b_2^2, \cdots,\ b_n^2$

with default

```
buck  b0  1.e-20
```

where

bn	specifies the order of the B_L flux solution and is optional, and
b_i^2	is the buckling for energy group i.

If a short list of b_i^2 is given, the last value in the list is used in the remaining groups. Thus one value is sufficient for a constant group buckling. A transport corrected P_0 calculation is actually performed instead of B_0.

Example:

```
buck  b1  .02
```

5.14. Search for Critical Buckling — search *

search $\left\{\begin{matrix}\textbf{off}\\ \textbf{on}\end{matrix}\right\}$ **bsq for** $k,\ \delta k$

with default

```
search off bsq for  1.  .0002
```

where

on, off	turns the search on or off and the default is 'no search',
k	is the value of k_{eff} required, and
δk	is the accuracy to which k is required.

The module searches for an eigenvalue λ such that λ times input buckling gives a k_{eff} of k. The initial buckling is set to 0.001 if it is not specified. The buckling in any following calculation is λB_g^2.

Example:

```
search on bsq for 1.02
```

5.15. Group Source — gsce

In any calculation which does not include a fissionable nuclide, a simple Maxwellian distribution with a temperature of 1.323 MeV (representing a ^{235}U fission spectrum) is used as the source for the group flux calculation. If this default is not satisfactory, a group source may be specified by

gsce $S_1, S_2, \ldots, S_n$

or

gsce *-t*

where S_i is the source in group i and a short vector is filled out with zero, and
t is the temperature, in eV, of the simple Maxwellian form of a fission spectrum.

The resulting fluxes are normalised to a total source of 1 neutron $cm^{-3}s^{-1}$. In fact, the source may be overwritten in this manner in any calculation.

5.16. Nuclide Fission Spectra — fstemp

The dependence of fission spectra on incident neutron energy is one quantity which is not included in the cross-section library. The library value of the fission spectrum of a nuclide is that appropriate for thermal calculations. Provision has, therefore, been made for fission spectra to be input. The form is

fstemp $name_1, t_1, name_2, t_2, \ldots$

where $name_i$ is the name of a library nuclide whose fission spectrum is to be altered, and
t_i is the temperature of the simple Maxwellian form of the fission spectrum in eV.

Example:

```
fstemp u235 1.323e+6
```

This changes the fission spectrum for ^{235}U to that used on the *aus/endf200g* library.

5.17. Bilinear Weighting Option — weight *

Flux weighting is normally used to average cross sections and to form printed reaction rates. If bilinear weighting is required, the entry

weight bilinear

should be included. The bilinear weighting formulae are the same as those used in EDITAR [Robinson 1986].

5.18. Type of Output Required — output *

output xs rr aus *lib* **flux geom zero p***n*

with default of

```
output aus 10 p0 flux geom
```

A selection may be made of any of the given parameters. The selection process is additive however, so **zero** must be used to stop any output option previously selected. The options have the meaning

xs print cross sections,
rr print reaction rates per unit source, or for bilinear weighting, print material worth components plus a few simple reaction rates,
aus write an AUS cross-section data pool on FORTRAN unit *lib*, and add to the STATUS data pool as necessary,
flux write an AUS FLUXB data pool for the main geometry mesh,

geom write an AUS geometry data pool, and
p*n* scattering matrices on the cross section data pool produced up to order P_n.

All **output** is for the materials on the **reqd** entry in condensed groups.

A series of cases may be performed with output on the same data sets. A composite cross-section data pool is formed automatically by MIRANDA.

Examples:

```
output xs rr
```

This adds print options to the standard option of writing all data sets.

```
output zero rr
```

This prints reaction rates and nothing else is produced.

5.19. Condensed Group Structure — groups *

groups n, $g_1, \cdots, g_{n+1}$

where n is the number of condensed groups,
g_1 is the first library group of condensed group 1, and
g_i for i > 1, is the last library group of condensed group i - 1.

If $g_1 < 1$, the g_i are taken to be group boundaries in lethargy units. In this case library groups with the nearest lethargy boundary are selected. Although the outer boundaries g_1 and g_{n+1} are specified, they must cover the full energy range of the cross-section library.

Example:

```
groups 20 1 10(10)200
```

5.20. Transport Cross Section Option — transport

Transport cross sections following a B_L calculation are normally adjusted to give the correct leakage in a diffusion theory calculation of a reactor. If this is not appropriate, the keyword required is

transport p1

The most likely reason for including this entry is that the cross sections are to be used in a global S_N transport calculation with P_0 scattering.

5.21. Specification of Output Nuclides — isotlib *

isotlib *liba* $name_1, \cdots, name_n$ **burnup p***n*

with default

```
isotlib  allmat  p-1 ,
```

which results in the nuclide data being added to the main output cross-section data pool,

where all quantities are optional,
liba is the FORTRAN unit on which a nuclide cross-section data pool is written and it may be the same as the main output cross-section data pool,
$name_i$ is a defined material whose constituent nuclides are to be produced, or is the keyword **allmat** which causes the constituent nuclides of all **reqd** defined materials to be produced,
burnup causes the definitions of the set of materials to be modified to include a small amount of all nuclides produced in that material during burnup, and
p*n* is the order of scattering matrices, with the default being no matrices at all.

The default values only apply if an **isotlib** entry is given. The option is turned off in the standard state.

The use of this entry enables nuclide reaction rate editing and burnup calculations to be performed within the AUS system. The connection between macroscopic materials and the nuclides of this cross-section data pool is maintained by information entered in the STATUS data pool by modules of the system.

An **isotlib** entry which includes the **burnup** specification is normally included in all MIRANDA calculations which form part of a burnup calculation. The entry must be included if the fission-product yields are to be modified (see **use** entry).

The entry

isotlib 0

turns off the option after it has been on in a previous case. A series of cases may be run with **isotlib** output on the same unit.

5.22. Inclusion of Additional Reactions — setreac

The *endfb6* library includes data for a number of additional reactions for many of the more important reactions. These reactions are identified by ENDF MT numbers, some of which are:

16	(n,2n)		
102	(n,γ)		
103	(n,p)	203	total proton production (H gas)
105	(n,t)	205	total tritium production
107	(n,α)	207	total alpha production (He gas)
444	total damage energy		

The additional reactions may be accessed within AUS by setting one of the output reactions to the desired MT number. Derived AUS cross section files still only have cross sections for nine reactions. The required input is:

setreac *irn mtn* $nuc_1, \cdots, nuc_n$

where *irn* is the AUS output reaction number (<10),
mtn is the MT value, and
nuc_i are nuclides on the library for which AUS reaction *irn* is to be set to MT *mtn* on the AUS output file.

A number of SETREAC entries can be included. The cross section will be set to zero for those nuclides which don't have cross sections given for *mtn*. Getting sensible resultant cross sections, particularly for mixtures, is the responsibility of the user.

Example:

```
setreac 9 444 fe cr ni
```

to set reaction 9 to total damage energy for the three main elements of steel.

5.23. Fission Energy Release — fer

The total energy release values for fissionable nuclides on the *endfb6* library have been set to the Q values from ENDF/B-VI plus an allowance for capture heating of 9 MeV [Robinson 1993]. Where this is not appropriate for a class of calculations, revised values may be calculated in some subsidiary calculations and then included using:

fer nuc_1 fer_1 , nuc_2 fer_2 , ...

where nuc_i is a fissionable nuclide on the library, and
fer_i is the fission energy release in MeV or joules.

5.24. Nuclide Regions in Output — reqdreg

reqdreg j_1 j_2 j_3 $name_1, \ldots, name_n$

with default

```
reqdreg 2 1 1
```

where the j_i take the values:

1	which means that cell average nuclide cross sections are produced, or
2	which means that a set of nuclide cross sections is produced for each resonance region in which the nuclide occurs and which is filled by a different material, and
$name_j$	are optional nuclide names for which multiple sets of cross sections are to be produced (n<20),

The three values are for fuel nuclides (atomic mass > 220), normal nuclides and fission products respectively. The nuclides referred to are the library nuclides which are on the **reqd** entry or which are generated from the **isotlib** entry.

The names of a set of nuclides which are produced are made distinct by combining the 6-character jobname given on the **head** entry with a 2-character resonance region number to form the 'source' component of the nuclide name.

5.25. Alternative Label — nucmod

The form of the material definition written by MIRANDA in the STATUS data pool is, for example,

```
fuel abcdefnn orig  =  u235 abcdefnn orig .003, u238 abcdefnn
orig .04
```

where `abcdef` is the *Jobname* on the **head** card,
nn is two characters which are blank or give a resonance region number,

and the input definition of `fuel` is

```
defn fuel  u235 .003  u238 .04
```

In some burnup applications, a common nuclide data pool may suffice for a number of cell types. This is made possible by specifying

nucmod *name*

where *name* is 1 to 6 characters which are used rather than *Jobname* on the right hand side of material definitions.

5.26. Resonance Options

resparm *nrespa*, ε_{r1}, ε_{r2}, ε_{r3}, *nangle*, *numer*, *multi*

with default

```
resparm  1  .001  .002  .001  8  0  1
```

where *nrespa* is the number of passes to be made through the set of light resonance nuclides for each group,
$\varepsilon_{r1}, \varepsilon_{r2}, \varepsilon_{r3}$ are the accuracy criteria of equations (5.6), (5.20) and (5.89), respectively, of Robinson [1977],
nangle is the order of Gaussian quadrature in the collision probability routine,
numer is a switch which, for a non-zero value, allows the numerical collision probability routine to be used in cylindrical geometry, and

multi is a switch whereby non-zero means that λ is mesh dependent and it is recommended that zero be used when there is more than one mesh interval per physical region. Weighting by volume and resonance nuclide concentration is used to form the average λ.

resopt m n

with default

```
resopt  0  0
```

where $m \neq 0$ results in an additional resonance overlap effect which depends on the average position of resonances in a group and the group resonance escape probability for other resonance reactions (Section 4.),
$n = 1$ if the Hill-Schaeffer λ method is to be replaced by a pseudo NR method for fissile nuclides, and
$n = 2$ if the pseudo NR method is to be used for all nuclides.

In the pseudo NR method, a λ value is applied to the resonance nuclide only; this allows conformity with the derivation of the subgroup parameters on the library but is otherwise an NR method.

efofp E_p

with default

```
efofp  5.e+4
```

where E_p is the energy in eV above which the removal correction factor $f_l(p_g)$ given in section 5.8 of Robinson [1977] is not applied.

pearls *ipearl*

with default

```
pearls  0
```

where *ipearl* takes the values

0 normal calculation,
1 a slowing down calculation in which the source term of the homogeneous flux calculation is an asymptotic slowing down source rather than a fission source, and
2 as for 1 but with a neutron mass of unity.

This entry is used for comparisons with the PEARLS code using a special purpose cross-section library which includes nuclide slowing-down distributions in place of fission spectra.

5.27. Flux Options

gflux $\phi_1, \cdots, \phi_{NG}$

or

gflux westcott r

with default

```
gflux westcott .08
```

where ϕ_i is a group flux estimate for the i^{th} library group,
NG is the number of groups on the library,
$r(>0)$ is the ratio of epithermal to thermal flux in a group flux estimate of Westcott form,
$r(<0)$ gives an estimated flux/lethargy of an integrated fission spectrum times $E^{|r|}$, and
$r = 0$ gives a fission spectrum.

The fission spectrum used in defining this flux estimate is that of the first fissionable nuclide included in the system. If there are no fissionable nuclides in the system, the fission spectrum is set to 1 in the first library group. Unless the flux calculation is bypassed, this flux estimate is used only to weight the fission spectra for use in the homogeneous flux calculation. Hence it is normally of no importance.

fluxparm *limfl*, *limsea*, *nscale*, ε_f

with default

```
fluxparm  40  10  3  1.e-4
```

where
- *limfl* is the limit on the number of iterations to converge the group flux calculation,
- *limsea* is not used at present,
- *nscale* is the number of broad groups used in scaling the thermal flux to improve convergence, and
- ε_f is the required flux accuracy.

If *limfl* = 0, a flux calculation is not performed and cross sections may be averaged over an input flux spectrum given by **gflux**.

5.28. Print Control of Intermediate Data — test

test n

with default

```
test  0
```

where
- $n = 0$ for a normal printout of input data, cross-section library contents, material table, k_{eff}, flux and reaction rates in the library groups
- $n = 1$ prints, additionally, the cross sections of resonance nuclides,
- $n = 2$ prints, additionally to 1, information on the resonance calculation and the flux convergence which is used in debugging,
- $n = -1$ prevents most normal printout, including the library contents if a **test** entry is placed first or after the **head** entry.

5.29. Write Geometry — write geomlib

write geomlib

The entry causes a geometry data pool to be written immediately and may be used to create a geometry data pool which is different from the geometry of the MIRANDA calculation.

5.30. Options for Photon Treatment*

The default option is to ignore any photon data on the cross-section library so that neutron-only calculations are not affected by the presence of those data. Treatment of photons is performed if a group structure for photon output is specified. There are three entries which control the photon treatment.

gamma n $g_1, \cdots, g_{n+1}$

where
- $|n|$ is the number of condensed photon groups,
- $n>0$ results in an output AUS cross-section data pool in which photon groups are treated as for neutron groups,
- $n<0$ results in an output AUS cross-section data pool in which photon groups are stored separately (normally such data are used as input to MIRANDA only), and
- g_i are photon group boundaries as for the **groups** entry.

When **gamma** is specified, the default calculation is a B_L calculation of the photon flux of the order specified on the **buck** entry, using a photon production source from the neutron flux calculation. As for the **gsce** entry for neutrons (Section 5.15), this default may be altered by using a **gasce** entry to specify the source in each photon group.

gaflux $\phi_1, \cdots, \phi_N$

or

gaflux -1

where either usage causes the B_L photon flux calculation to be bypassed,
ϕ_i specifies the flux in the i^{th} photon group, and
-1 causes a constant in energy flux to be used.

A blank **gaflux** entry restores the default B_L calculation.

The default order of P_n scattering expansion for photon interaction data on output is the same as that for neutrons. To modify this default, **gap** n is included on the **output** entry.

5.31. Begin Calculation — start *

start

The entry causes the calculation which has been set up by the previous entries to begin.

5.32. Control of STATUS Data Pool

read status

This entry causes the STATUS data pool to be read immediately. This will set up a succeeding calculation which is specified by the STATUS data pool.

statoff

This entry causes nothing to be written on either STATUS data pool. It is useful for perturbations on a burnup calculation.

The normal usage would be

(a) burnup to a required time,
(b) perform perturbed calculations with **statoff**,
(c) perform standard calculations and continue burnup.

5.33. Terminate Calculation — stop *

stop

This entry causes termination of the module.

5.34. Example

A sample of input for a lattice calculation using the modules MIRANDA, ANAUSN [Clancy 1982] and EDITAR [Robinson 1986] is given below. The calculation includes

(i) MIRANDA - 3-region resonance calculation, B_3 flux calculation and condensation to 26 groups;
(ii) ANAUSN - 26-group k_∞ S_6 spatial calculation with P_1 scattering; and
(iii) EDITAR - edit of ANAUSN fluxes, B_3 calculation to give final k_{eff} and two-group reaction rates.

```
aus port2 <<'eof'
*dd1
step *
* TRX1 Lattice Experiment as documented in
* "CSEWG Benchmark Specifications" - BNL 19302 (ENDF-202).
* An hexagonal boundary is used in the resonance calculation.
      link miranda
      link anausn(1,3)
      link editar(1,4)
      end
stop
*dd2
head trx1 ref endf-202
defn fuel u235 6.253-4 u238 4.7205-2
defn void o 1.-5
defn clad al .06025
defn mod h2o .03338
reqd fuel void clad mod u235 u238
rm 0 5*.0983 .0127 .0711 5*.0745836 0
reg 1(1)5 fuel 6 void 7 clad 8(1)12 mod
resreg 6 .4915 .0838 .372918 -1 smear 5*1 2 2 5*3
buck b3 5.7-3
output p3
groups 26 -1 .5(0.5)2.5(1.5)10. 12.5 14.3 15.3
16.2(0.4)20.2 21. 23.
start
stop
*dd3
trx
aus nl 1 nsn 6 end
*dd4
buck b3 5.7-3    search off
groups 2 1 15 26
output rr start
```

6. REFERENCES

Bennett, N.W., Pollard, J.P. [1967] - SCAN - a free input subroutine for the IBM360. AAEC/TM399.

Chiarella, C. [1971] - PEARLS - a code for the solution of the neutron slowing down equations in multiregion lattices of resonance absorbers. AAEC/E213.

Clancy, B.E. [1982] - ANAUSN - a one-dimensional multigroup SN transport theory module for the AUS reactor neutronics system. AAEC/E539.

Clark, H.K. [1982] - Subcritical limits for Uranium-235 systems. Nucl. Sci. Eng. **81**, 351-378.

Doherty, G. [1969] - Some methods of calculating first flight collision probabilities in slab and cylindrical lattices. AAEC/TM489.

Doherty, G., Robinson, G.S. [1985] - MCRP - a Monte Carlo resonance program for neutrons slowing down in single rod and rod cluster lattices. AAEC/E618.

Robinson, G.S. [1975] - AUS burnup module CHAR and the associated STATUS data pool. AAEC/E372.

Robinson, G.S. [1977] - AUS module MIRANDA - a data preparation code based on multiregion resonance theory. AAEC/E410.

Robinson, G.S. [1979] - Approximate first collision probabilities for neutrons in cylindrical and cluster lattices. AAEC/E465.

Robinson, G.S. [1981] - Notes on the AAEC version of XLACS2. ORNL/RSIC/PSR-182.

Robinson, G.S. [1985] - A comparison of neutron resonance absorption in thermal reactor lattices in the AUS neutronics code system with Monte Carlo calculations. AAEC/E614.

Robinson, G.S. [1986] - EDITAR - a module for reaction rate editing and cross-section averaging within the AUS neutronics code system. AAEC/E621.

Robinson, G.S. [1987] - A guide to the AUS modular neutronics code system. AAEC/E645

Robinson, G.S. [1993] - Generation and validation of a cross section library based on ENDF/B-VI for the AUS neutronics code system. ANSTO/E712.

Table 1. Neutron Group Boundaries for AUS *endfb6* Library

Group	Lower Energy (MeV)	Lethargy	Group	Lower Energy (MeV)	Lethargy	Group	Lower Energy (keV)	Lethargy
0	15.488	-0.43750						
1	15.012	-0.40625	34	2.231	1.5000	67	283.68	3.5625
2	14.550	-0.37500	35	2.096	1.5625	68	266.49	3.6250
3	14.102	-0.34375	36	1.969	1.6250	69	250.35	3.6875
4	13.668	-0.31250	37	1.850	1.6875	70	235.18	3.7500
5	13.248	-0.28125	38	1.738	1.7500	71	220.93	3.8125
6	12.840	-0.25000	39	1.632	1.8125	72	207.54	3.8750
7	12.062	-0.18750	40	1.534	1.8750	73	194.97	3.9375
8	11.331	-0.12500	41	1.441	1.9375	74	183.16	4.0000
9	10.645	-0.06250	42	1.353	2.0000	75	172.06	4.0625
10	10.000	0.00000	43	1.271	2.0625	76	161.63	4.1250
11	9.394	0.06250	44	1.194	2.1250	77	151.84	4.1875
12	8.825	0.12500	45	1.122	2.1875	78	142.64	4.2500
13	8.290	0.18750	46	1.054	2.2500	79	125.88	4.3750
14	7.788	0.25000	47	0.990	2.3125	80	111.09	4.5000
15	7.316	0.31250	48	0.930	2.3750	81	98.04	4.6250
16	6.873	0.37500	49	0.874	2.4375	82	86.52	4.7500
17	6.456	0.43750	50	0.821	2.5000	83	76.35	4.8750
18	6.065	0.50000	51	0.771	2.5625	84	67.38	5.0000
19	5.698	0.56250	52	0.724	2.6250	85	59.46	5.1250
20	5.353	0.62500	53	0.681	2.6875	86	52.48	5.2500
21	5.028	0.68750	54	0.639	2.7500	87	46.31	5.3750
22	4.724	0.75000	55	0.601	2.8125	88	40.87	5.5000
23	4.437	0.81250	56	0.564	2.8750	89	36.07	5.6250
24	4.169	0.87500	57	0.530	2.9375	90	31.83	5.7500
25	3.916	0.93750	58	0.498	3.0000	91	28.09	5.8750
26	3.679	1.00000	59	0.468	3.0625	92	24.79	6.0000
27	3.456	1.06250	60	0.439	3.1250	93	21.87	6.1250
28	3.247	1.12500	61	0.413	3.1875	94	19.30	6.2500
29	3.050	1.18750	62	0.388	3.2500	95	17.04	6.3750
30	2.865	1.25000	63	0.364	3.3125	96	15.03	6.5000
31	2.691	1.31250	64	0.342	3.3750	97	13.27	6.6250
32	2.528	1.37500	65	0.321	3.4375	98	11.71	6.7500
33	2.375	1.43750	66	0.302	3.5000	99	10.33	6.8750

Table 1. (continued)

Group	Lower Energy (eV)	Lethargy	Group	Lower Energy (eV)	Lethargy	Group	Lower Energy (eV)	Lethargy
100	9118.82	7.000	134	13.7096	13.50	168	0.18602	17.8
101	8047.33	7.125	135	10.6770	13.75	169	0.16832	17.9
102	7101.74	7.250	136	8.3153	14.00	170	0.15230	18.0
103	6267.27	7.375	137	6.4760	14.25	171	0.13781	18.1
104	5530.84	7.500	138	5.0435	14.50	172	0.12469	18.2
105	4880.95	7.625	139	3.9279	14.75	173	0.11283	18.3
106	4307.43	7.750	140	3.0590	15.00	174	0.10209	18.4
107	3801.29	7.875	141	2.7679	15.10	175	0.09237	18.5
108	3354.63	8.000	142	2.5045	15.20	176	0.08358	18.6
109	2960.45	8.125	143	2.2662	15.30	177	0.07563	18.7
110	2612.59	8.250	144	2.0505	15.40	178	0.06843	18.8
111	2305.60	8.375	145	1.8554	15.50	179	0.06192	18.9
112	2034.68	8.500	146	1.6788	15.60	180	0.05603	19.0
113	1795.60	8.625	147	1.5191	15.70	181	0.05070	19.1
114	1584.61	8.750	148	1.3745	15.80	182	0.04587	19.2
115	1398.42	8.875	149	1.2437	15.90	183	0.04151	19.3
116	1234.10	9.000	150	1.1254	16.00	184	0.03756	19.4
117	961.12	9.250	151	1.0183	16.10	185	0.03398	19.5
118	748.52	9.500	152	0.9214	16.20	186	0.03075	19.6
119	582.95	9.750	153	0.8337	16.30	187	0.02782	19.7
120	454.00	10.000	154	0.7543	16.40	188	0.02518	19.8
121	353.58	10.250	155	0.6826	16.50	189	0.02278	19.9
122	275.36	10.500	156	0.6176	16.60	190	0.02061	20.0
123	214.45	10.750	157	0.5588	16.70	191	0.01865	20.1
124	167.02	11.000	158	0.5057	16.80	192	0.01688	20.2
125	130.07	11.250	159	0.4575	16.90	193	0.01527	20.3
126	101.30	11.500	160	0.4140	17.00	194	0.01382	20.4
127	78.89	11.750	161	0.3746	17.10	195	0.01250	20.5
128	61.44	12.000	162	0.3389	17.20	196	0.00758	21.0
129	47.85	12.250	163	0.3067	17.30	197	0.00460	21.5
130	37.27	12.500	164	0.2775	17.40	198	0.00279	22.0
131	29.02	12.750	165	0.2511	17.50	199	0.00169	22.5
132	22.60	13.000	166	0.2272	17.60	200	0.00001	27.6
133	17.60	13.250	167	0.2056	17.70			

Table 2 Photon Group Boundaries

Group	Lower Energy (MeV)	Group	Lower Energy (keV)
0	20.0		
1	14.0	20	1000
2	12.0	21	800
3	10.0	22	700
4	8.0	23	600
5	7.5	24	512
6	7.0	25	510
7	6.5	26	450
8	6.0	27	400
9	5.5	28	300
10	5.0	29	200
11	4.5	30	150
12	4.0	31	100
13	3.5	32	70
14	3.0	33	60
15	2.5	34	45
16	2.0	35	30
17	1.66	36	20
18	1.5	37	10
19	1.33		

Table 3. Nuclides in the AUS *endfb6* Library

Name	Source		*Pl* order	Temperature range	Resonance data?	Photon data?
zz999	njoy	orig	-			No
ch2	endfb6	m020	7	296		
h2o	endfb6	m010	7	296-1000		
h0zrh	endfb6	m010	7	296-1000		
d2o	endfb6	orig	7	296-1000		
he3	endfb6	m010	3	296		No
li6	endfb6	m010	7	296		
li7	endfb6	orig	7	296		
be	endfb6	orig	7	296-1000		
b10	endfb6	m010	7	296		
b11	endfb6	orig	7	296		
c	endfb6	m010	7	296-1600		
n	endfb6	orig	7	296		
o	endfb6	orig	7	296-3000		
f	endfb6	orig	3	296		No
na	endfb6	m010	7	296	Yes	
mg	endfb5	njoy	3	296		
al	endfb5	njoy	7	296	Yes	
si	endfb5	njoy	7	296		
cl	endfb5	njoy	3	296		
ca	endfb5	njoy	7	296		

Name	Source		Pl order	Temperature range	Resonance data?	Photon data?
ti	endfb5	orig	3	296		
v	endfb6	orig	3	296		
cr	endfb6	m040	7	296	Yes	
mn	endfb6	orig	3	296	Yes	
fe	endfb6	m040	7	296	Yes	
co	endfb6	orig	3	296		
ni	endfb6	m050	7	296	Yes	
cu	endfb6	orig	3	296		
ga	endfb5	njoy	1	296		No
zr	endfb6	m010	7	296-1000	Yes	
zr0zrh	endfb6	m010	7	296-1000	Yes	
nb	endfb6	orig	3	296		
mo	endfb5	orig	3	296		
cd	endfb5	m010	3	296		
ba	endfb5	njoy	1	296		
gd	endfb5	njoy	1	296		No
er166	endfb6	orig	1	296		No
er167	endfb6	orig	1	296		No
lu176	endfb5	njoy	1	296		No
au	endfb6	m010	1	296		No
pb	endfb6	m010	7	296	Yes	
bi	endfb6	orig	7	296	Yes	
u232	pu236001	orig	0	296		No
u234	endfb6	orig	3	296-900	Yes	
u235	endfb6	m010	7	296-2100	Yes	
u236	endfb6	orig	3	296-900	Yes	
u237	endfb6	orig	3	296		
u238	endfb6	m010	7	296-2100	Yes	
np237	endfb6	m010	3	296		
np239	endfb6	orig	3	296		
pu236	endfb5	njoy	3	296		No
pu238	endfb6	orig	3	296		
pu239	endfb6	orig	3	296-2100	Yes	
pu240	endfb6	m010	3	296-2100	Yes	
pu241	endfb6	m010	3	296-2100	Yes	
pu242	endfb6	orig	3	296-2100	Yes	
am241	endfb6	orig	3	296		
am242m	endfb6	m010	3	296		
am242	endfb6	m010	3	296		
am243	endfb6	orig	3	296		
cm242	endfb6	orig	3	296		
cm243	endfb5	njoy	3	296		
cm244	endfb5	njoy	3	296		
cm245	endfb6	orig	3	296		

Table 4. Fission Products in the AUS *endfb6* Library

	Mass Numbers and possible Isomeric State for each element	
Element	Normal Set	Extended Set
ge		72 73 74 76
as		75
se		76 77 78 80 82
br		81
kr	83 85	82 84 86
rb		85 87
sr	90	86 88 89
y		89 91
zr	95	90 91 92 93 94 96
nb	95	
mo	95	96 97 98 99 100
tc	99	
ru	100 101 102 103 105 106	104
rh	103 105	
pd	104 105 106 107 108	110
ag	109	111
cd	113	110 111 112 114 115m 116
in		115
sn		116 117 118 119 120 122 123 124 126
sb		121 123 124 125
te		122 123 124 125 126 127m 128 129m 130 132
i	135	127 129 131
xe	131 133 135	128 130 132 134 136
cs	133 134 135 137	136
ba		134 136 137 138 140
la		139 140
ce	144	140 141 142 143
pr	143	141
nd	143 144 145 146 147 148	142 150
pm	147 148m 148 149 151	
sm	147 149 150 151 152	148 153 154
eu	153 154 155 157	156
gd	157	154 155 156 158 160
tb		159 160
dy		160 161 162 163 164
ho		165

ANAUSN - ONE-DIMENSIONAL MULTIGROUP SN TRANSPORT THEORY MODULE

1. INTRODUCTION

ANAUSN is a general purpose, one-dimensional discrete ordinate transport theory program which has access to AUS data pools. The program is an implementation of Carlson's discrete ordinate S_N method for the solution of multi-group transport problems in one-dimensional geometries - plane, cylindrical and spherical [Carlson 1963]. ANAUSN can be used to perform fixed source, reactivity and a variety of criticality search calculations.

ANAUSN can be operated as a module in the AUS scheme or as a stand-alone program. As a module in the AUS system it serves two functions:

a. In large heterogeneous reactor systems, ANAUSN treats the single cell multiplication problem to produce tentative group fluxes which can be used to homogenise the cell and to provide the few group cross sections for use with multi-dimensional diffusion theory codes. The ANAUSN calculation produces a solution of an eigenvalue problem, with the eigenvalue being the cell fission multiplication factor k_∞ .

b. For systems which can be adequately represented in one dimension but for which the diffusion theory approximation is unacceptable, ANAUSN provides the necessary transport theory solution. The calculation may be the eigenvalue problem, equivalent to that carried out for a cell calculation, which determines the system fission multiplication factor k_{eff}. Optionally, the code may be asked to vary some system parameter and search for a value which makes the multiplication factor take some specified value - usually 1.0. A quite separate calculation is that for source problems, in which a fixed source (of neutrons) is being released in a sub-critical system and the determination of steady state group fluxes is required. In all cases, either the normal or adjoint transport equation can be solved.

The basic solution strategy is that laid down in the original S_N codes of Carlson [Carlson and Bell 1958; Carlson et al. 1960] and followed essentially unchanged in all S_N codes since that time. Consequently, it is inappropriate to rehearse the theory of the S_N treatment and the terminology and nomenclature in the Carlson [1963] account are retained, assuming that the reader is familiar with them.

The task of developing a new code was undertaken for two reasons: to provide efficient access to the AUS scheme data pools and to have the opportunity of implementing improvements in numerical techniques for producing converged solutions. Development of the code was begun in 1972 and a first version was installed in the AUS scheme in 1973. A number of revisions have been made as experience with the code was accumulated. This report replaces the original edition [Clancy 1982] which described the code version installed in the 1981 version of the AUS scheme. The current version of ANAUSN is written completely in FORTRAN 77.

2. ZONING AND PROBLEM SPECIFICATION

Specification of the material layout through user-supplied input data and most of the acceleration procedures depend on the concept of a material zone; this is simply a set of contiguous mesh intervals which are occupied by a single material. When defining the material layout, the user's data must assign each interval to a particular zone and then provide a material number to be associated with each of the zones.

Depending on the problem type, the number of mesh intervals and the number of material zones, ANAUSN may subdivide the zones even further unless **prel** is used. For example, a material zone with, say, fifteen mesh intervals may be subdivided into three or four zones for rebalance purposes even though only one material is present throughout the fifteen mesh intervals. Except for cell calculations, ANAUSN attempts to make the average number of mesh intervals per zone equal to four.

The rebalance techniques take the flux iterates calculated as a result of inner iterations and multiply these by scaling factors computed in an attempt to accelerate the iteration procedure. A single scale factor is sometimes computed for each zone, and all fluxes in that zone are multiplied by that factor. With very many mesh intervals per zone, the acceleration procedure is less efficient whereas too few mesh intervals per zone may produce instabilities in the acceleration factors and result in wasted computations.

3. REPORT CONVENTIONS

Conventions are used in this report to make clear, for example, the distinction between a keyword like **eps** and the number EPS which is read in as soon as the keyword is sensed. The adopted conventions follow:

- module names are given in UPPER CASE,
- types of data file are given in UPPER CASE,
- variable names are given in UPPER CASE,
- examples and defaults are given in `constant width type`,
- keywords (which must often be followed by numerical data items) are given in **bold** and the exact words must be used, the data items themselves are referred to by using the same word in non-bold UPPER CASE.

4. INPUT

Except for the title, all input data are read by SCAN which searches input records for approved keywords. Only the first 72 characters of each input record are processed. The keywords are used to identify any string of numeric data items which follow. The data to be provided for a particular case fall logically into five data blocks. Unless specifically stated, the ordering of data items within blocks is immaterial.

4.1 Data Block 1

One title record with the requested maximum CPU time for the case in integer minutes occupying columns 69-72. If this time is left blank or is greater than the remaining job time the maximum available job time is used.

4.2 Data Block 2

This block specifies a set of prelude numbers which determine the problem type and size. They are used to determine allocation of storage arrays for the case. One of two keywords must begin this data block; they are **prel** or **aus**. The first option is

prel (short for prelude) Here and for all keywords only the first four letters are significant. If given, this keyword must be followed by the following 16 integer parameters:

ID	The case number.
IADJ	0 for normal transport calculation, 1 for an adjoint calculation and 2 if both normal and adjoint calculations are to be performed.
NL	0 if isotropic scattering theory is wanted, 1 if P_1 scattering is wanted, 2 if P_2 scattering is wanted, etc.
NSN	The order of S_N treatment and so must be even.
IEVT	The type of problem: 0 for fixed source calculations; 1 for k_{eff} calculations; 2 for an eigenvalue search (for k_{eff} = REQK) on α/v absorption, where α is the eigenvalue, v is the group velocity; 3 for an eigenvalue search with the whole reactor radius varied, all intervals being varied uniformly - the reactor radius is the eigenvalue; 4 for an eigenvalue search with the size of material zones being varied individually; 5 for an eigenvalue search on material concentrations; 6 for an eigenvalue search by varying a single transverse buckling B^2 and adding a pseudo-absorption of D. B^2 in each interval of each group - here D is the material group diffusion coefficient and the eigenvalue is the value of B^2.
NMIX	The number of entries to be entered in each mixing table for cross sections.
JOM	The geometry number, 0 = slab (plane) geometry, 1 = cylindrical geometry, 2 = spherical geometry.
IBL	The left or inner boundary condition on the angular fluxes; 0 implies a free (vacuum) boundary, 1 implies a perfect specular reflection boundary, 2 implies a periodic boundary condition and 3 implies a white reflective boundary.
IBR	The right or outer boundary condition with the same meanings as IBL.
NZ	The number of material zones.
NMAT	The total number of materials for which cross sections will be supplied through data or by mixing.
NI	The number of mesh intervals.
NQV	The number of mesh intervals which contain a fixed isotropic volume source.
NQA	The number of mesh intervals into which fixed angular shell sources are directed.
NG	The number of groups for the problem.
MAXOUT	The maximum number of outer (power) iterations to be used for convergence.

After reading these 16 values, ANAUSN goes on to data block 3.

As an alternative, the second option may be invoked to determine the prelude numbers; here the user enters the keyword **aus** instead of **prel**. This causes the code to search an AUS geometry data pool GM1 to determine values for the parameters

JOM, IBL, IBR, NZ, NMAT, NI

while a cross section data pool XS1 is searched to determine values for the parameters NL and NG.

Default values for the remaining parameters are assigned as

`id=1`	for first case within the task, 2 for second case etc.,
`iadj=0`	0 implies normal transport calculation,
`nsn=4`	S_4 treatment,
`ievt=1`	a k_{eff} calculation, and
`nmix=0`	0 hence no cross section mixing tables.

Any of these default values can be overridden by entering the parameter name as a keyword and following it with the new value. Thus to specify an eigenvalue search on outer radius using S_{16} theory, the data for this block could be

```
aus ievt 3 nsn 16
```

The explicit entry of values is essential for a job in which the AUS data pools GM1, XS1 have not been created earlier. Note that if ANAUSN is run without an AUS geometry data pool then the following parameters need to be set

JOM, IBL, IBR, NSN, NI, NZ, NMAT

The parameter NL should be set in any ANAUSN run in which the MLZ values are not the same as on the cross section data pool.

4.3 Data Block 3

This block of data may be used to specify certain parameters which further define the problem or else give directives to the code. Again the input procedure consists of entering the keyword which may have to be followed by an input value. Default conditions have been set for all quantities which may be specified here and, in most cases, the block may be omitted. The keywords, corresponding data items, their function and the default values are listed below.

ev followed by 1 real number EV. First guess for eigenvalue searches.

ev2 followed by 1 real number EV2. Second guess for eigenvalue searches.

Default EV and EV2 values selected by the code depend on the type of search.

eps followed by 1 real number EPS. The accuracy in k_{eff} and/or the eigenvalue; default is 5×10^{-4}.

epsp followed by 1 real number EPSP. The relative accuracy required in all scalar fluxes; default is 5×10^{-4}.

The actual accuracy in k_{eff} may be worse than EPS and EPSP in contrast to POW3D where it is much better. This is because convergence is assumed when the differences for successive outer iterations in values of k_{eff} and flux are less than EPS and EPSP respectively.

reqk followed by 1 real number REQK. The k_{eff} value which searches try to achieve; default REQK = 1.0.

alfa followed by 1 real number ALFA. An ALFA/v absorption is added to all group cross sections; default ALFA = 0.0 .

bsq followed by 1 real number BSQ. An absorption $BSQ/(3 \times \Sigma_{tr})$ is added; default BSQ = 0.0

maxi followed by 1 integer MAXI. The maximum number of inner iterations allowed per group per outer iteration; default is 100. The user may be able to save time in global calculations by setting MAXI to approximately 10.

insc followed by 1 integer INSC. Controls the type of rebalance acceleration used after each inner iteration. If INSC = 0, no rebalance is applied, if INSC = 1, Bell's simple scaling, reported by Carlson and Bell [1958], is used so that the reactor fluxes are rebalanced as a whole after each inner, and if INSC >1 then for INSC inner iterations, a region rebalance will be applied every third iteration and simple scaling used on any other inner iterations. Default value is 30.

igsc followed by 1 integer IGSC. Controls the method of group rebalance used in problems with upscatters. If IGSC = 0, no group rebalance is applied; if IGSC = 1, the fluxes in the thermal upscatter groups are rebalanced globally after each outer iteration using a different scale factor for each upscatter group but applying it uniformly throughout the reactor. If IGSC > 1, the global rebalance is used after most outer iterations, but periodically a separate factor is evaluated for each rebalance zone and each upscatter group. The frequency of application of this is determined by the next parameter. The default value is 2 unless a cell calculation is being performed when the default is 1.

maks followed by 1 integer MAKS. The frequency of application of group and zone rebalance in upscatter problems. Default value is 3. For problems with no upscatters, these last two parameters are irrelevant.

The remaining keywords in this block serve as directives and no numeric data items follow them.

diam forces the code to use the 'diamond difference' scheme when calculating angular fluxes, even when this produces negative angular fluxes, whereas

step forces the code always to use the 'step difference' scheme. Step and diamond schemes were defined by Carlson [1963]. The default scheme for the code is to use the diamond scheme except where it results in a negative angular flux, in which case the step scheme is used for that mesh interval and angular direction.

pn directs the code to choose the angular segmentation weights and ordinates to be identical with those corresponding to Legendre quadrature integration over the angular range $-1 < \mu < 1$, whereas

dpn directs the code to use weights and ordinates corresponding to separate Legendre quadratures over the angular intervals $-1 < \mu < 0$ and $0 < \mu < 1$.

The default scheme uses P_N quadrature for spherical geometry, DP_N quadrature for plane geometry (so as to mitigate any thin slab effect in that geometry) and the equal weight quadratures of Brissenden as implemented in the WDSN code [Francescon 1963] for cylindrical geometry.

The next set of directive keywords in this block each direct the code to write a corresponding AUS data pool appropriate to the reactor as defined at the end of the case. Thus

wgm1		GM1	
wxs1		XS1	
wxs2	request data pool	XS2	to be written
wxs3		XS3	
wfl2		FL2	

In a default condition none of the data pools are written unless the keyword **aus** is detected anywhere in the input stream. In this event the FL2 data pool is written.

The final set of directive keywords either causes certain arrays to be printed or else stops them from being printed. The general default philosophy is that the user who specified **prel** as the first keyword will have all the arrays printed (with the exception of the angular fluxes) whereas the user who specified **aus** will get none of the arrays. The keywords are

wrxs	to have the cross sections printed, **noxs** to delete this
wrch	to have the fission spectra printed, **noch** to delete this
wred	to have input vectors printed, **noed** to delete this
wrba	to have zone balance table printed, **noba** to delete this
wrgb	to have each group balance table printed, **nogb** to delete this
wrba wrgb	to have zone balance tables for each group printed, **noba nogb** to delete this
wraf	to have group angular fluxes printed, **noaf** to delete this
wrsf	to have scalar fluxes printed, **nosf** to delete this

4.4 Data Block 4

Data within this block consist of keywords, each of which is followed by arrays of data items. If the keyword **aus** was given in data block 2, the code will remember this fact and search the appropriate data pools to find the eight arrays R, MZI, MNZ, MLZ, CHI, VEL, SF and XS before interpreting any keywords in the present block. The keywords together with the length, type and meaning of the arrays are given below:

r followed by NI + 1 real numbers. The positions R_i of the reactor mesh points, where R_1 is the left or inner boundary and R_{NI+1} is the right or outer boundary. Instead of supplying the mesh point positions in this form the user may give

dr followed by NI reals. The mesh interval spacings, in this case R_1 is set to zero. Units for R or DR are cm.

mzi followed by NI integers. The material zone numbers for each mesh interval starting at the left-most one.

mnz followed by NZ integers. The material numbers for each of the NZ zones.

mlz followed by NZ integers. The order of P_L scattering which applies for each zone in turn. The default values are zero if the GM1 data pool is not given.

chi followed by NG reals. The fission spectrum for all materials.

vel followed by NG reals. The group velocities. Normal units are cm/shake ($10^8 \mathrm{cm\,s^{-1}}$). This vector should be entered if IEVT = 2 or if ALFA $\neq$ 0.0.

xs followed by any number of cross section blocks. To enter a block of data for the P_L component of material number M, the data consist of the integers M,L followed by the cross sections in modified WDSN layout. As many cross section blocks as desired can be entered here in any order; the code will continue to read them until a new keyword is sensed. Unless modified by the mixing tables, these are nominally macroscopic cross sections and the usual units are $\mathrm{cm^{-1}}$. The WDSN layout used for a P_0 component consists of NG sets of numbers where one set for each group starts with the highest energy group. The set for group g, say, would consist of the following:

(i) Two integers LSS and LV : the position in the group vector of the self-scatter cross section and the length of the group vector.
(ii) Two reals, Σ_{tr} and $\nu\Sigma_f$: the group transport cross section and the fission emission cross section.
(iii) the group vector containing LV reals : the first of which is the group absorption cross section and the remainder are the group outscatter cross sections - one of which is the group self-scatter cross section. The group transport cross section read-in is only used in problems for which a transverse leakage is significant, i.e. BSQ $\neq$ 0.0 or IEVT = 6 and, in either case, this transverse leakage is modelled in the code by adding a term BSQ/(3 $\times \Sigma_{tr}$) to each group total cross section and hence to the implied group absorption cross section.

For a P_L component cross section block with L > 0, the block layout is identical to that for L=0 except that Σ_{tr}, Σ_{abs} and $\nu\Sigma_f$ must be zero. The P_L cross section components, $\sigma^L_{gg'}$, are defined so that the angular scattering source in group g, direction μ is

$$S_g(\mu) = \frac{1}{2}\sum_{g'}\sum_{L=0}^{NL}(2L+1)\sigma^L_{gg'}P_L(\mu)\int_{-1}^{1}P_L(\mu')\phi_{g'}(\mu')d\mu'$$

and ANAUSN inserts the (L+1) factor when computing this source.

dc followed by NM reals. The direction cosines for the discrete directions of the S_N method. Here NM = NSN+1 for plane and spherical geometry, but NM = NSN×(NSN+4)/4 for cylindrical geometry.

w followed by NM reals. The corresponding S_N direction weights. The weights entered as data are normalised again by the code. Default values for the DC and W arrays are available.

fd followed by NI reals. A first guess for the fission density in each interval. Default values are generated if this array is not given explicitly.

sf followed by NI×NG reals. A first guess for the scalar flux in each interval for each group in turn. Again, the code will generate a default array if necessary.

mqv followed by NQV integers. Only entered for source problems with NQV > 0. The array lists the interval numbers which contain a fixed isotropic source.

qv followed by NG×NQV reals. The fixed isotropic source density spectra for each of the NQV source intervals in turn. Units are particles/cm^3/s.

mqa followed by NQA integers. Only entered if NQA > O. The array lists the interval numbers into which fixed angular dependent shell sources are directed.

qa followed by NM×NG×NQA reals. The shell source densities ordered by S_N directions, by group and by interval. If, for example, the first element of the MQA array is MQA(1) = 3, a non-zero element of the QA array such as QA(m,g,l) = q means that the group g angular flux in the discrete direction m is increased by an amount q over that leaving the 'previous' interval. If the direction cosine μ_m is positive, this previous interval will be interval 2, but if μ_m is negative, the previous interval will be interval 4.

zmod followed by NZ reals. Only entered if IEVT = 4; in this case the material zone thicknesses will be expanded or contracted according to this array of zone modifiers. Thus if mesh interval i (of thickness δ_i according to the original R or DR array) lies in zone j, its thickness when the eigenvalue has a value E will be set to $\delta_i[1 + E.ZMOD_j]$.

mte followed by NMIX integers,

mtc followed by NMIX integers, and

atd followed by NMIX reals. The mixing tables which can only be entered if NMIX > 0 and must be entered if IEVT = 5. The tables are interpreted by ANAUSN before iterations start and, if IEVT = 5, again after each estimate of k_{eff} has been made with the current eigenvalue. The four mixing rules are shown below, where k and m are positive integers and x is a real:

MTE	MTC	ATD	
0	± m	x	means multiply all cross sections for material m by x.
m	± m	0.0	means multiply all cross sections for material m by the current eigenvalue.
k	± m	x	means add a multiple x of the cross sections for material k to those for material m.
0	-m	0	means zeroise material m (if necessary before using other mixing rules).

All cross section mixing can be done with these rules. A negative entry in the MTC array means that the corresponding operation is only performed on the first interpretation of the mixing table. This provides a means of converting microscopic data to macroscopic form or, alternatively, of using a standard material cross section block but using a density or voidage factor with it.

matd followed by NZ integers. The material numbers to be used in getting D for D.BSQ.

gbsq followed by NG reals. Group bucklings. D.GBSQ absorption is added everywhere.
In the original version of ANAUSN [Clancy 1982] transverse leakage was treated by inclusion of a D.BSQ 'absorption' term. A group independent buckling could be entered in the data and the code would, for each material zone, calculate the diffusion coefficient $D = 1/3.\Sigma_{tr}$ of the material in the zone. The current version of ANAUSN still honours data entered as before. However the treatment can be changed, if the user wishes, in either or both of two ways (keywords **gbsq matd**). Group dependent bucklings, GBSQ, may be entered and/or the diffusion coefficient to be used with a buckling in each zone can be calculated from the Σ_{tr} of a material whose number can be specified (MATD) in the input data. If this specification is not given then the cross section data for the material actually occupying the zone is used in the calculation.

4.5 Data Block 5

This consists simply of the keyword **end** and signals the completion of data for the case.

4.6 Minimal Input

A special option is provided for the AUS user who has set up cross section and geometry data pools XS1 and GM1 to define the system and who wishes simply to have a calculation of multiplication factor k_{eff} made for that system. If the user supplies an empty data set as input, the code senses this and generates all necessary input. The same result could have been achieved by supplying as data the following records:

```
title record for the case
aus end
```

5. OUTPUT

5.1 AUS Data pools

If the code is executed within the AUS scheme, a FLUXB data pool is always written at the termination of a case - whether the case has been converged or not. Termination without convergence will occur if the maximum allowable number of outer iterations has been reached, or if the maximum allowable time for the job seems likely to be exceeded. In the latter situation, the data pool can be used as a flux guess to restart the calculation during another job. Data pools for geometry and cross sections will be written if the case input requests them.

5.2 Printed Output

Printed output for a case which appears on FORTRAN logical unit 6 can be brief or very full.

5.2.1 Minimal printed output

The minimum output consists of a marked listing of the case input (the marking being a statement of the keywords processed), a description of any AUS data pools from which data are taken, a monitor print for each outer iteration, a statement of any AUS data pools written after the case is terminated, and case timing information which is self-explanatory.

The monitor line is of interest mainly with problems for which convergence is difficult to achieve. The items printed in the monitor line are

- the outer iteration number;
- the cumulative count of inner iterations;
- the present eigenvalue estimate;
- the total fission rate in the system for this outer iteration (For eigenvalue problems the fluxes are always normalised so that there is exactly one fission neutron released; this monitor item is then the current estimate of the fission multiplication factor k_{eff}.);
- the overall neutron balance for the system which should be exactly 1.0;
- λ which, for eigenvalue problems, is the ratio of successive fission rates, but for fixed source problems is the ratio of successive estimates of fixed source plus fissions plus upscatter reactions (In both cases this should be 1.0 for a converged problem.);
- the maximum proportional change in any group scalar flux in any interval during the last outer iteration;
- the group and the interval in which this flux change occurred;
- an estimate of the dominance ratio when Chebychev acceleration of the fission density is being used;
- the maximum proportional change in fission density during the last outer iteration;
- the maximum and minimum scale factors used in the group rebalance for this outer iteration; and
- an indicator to show the form of rebalance applied on this outer iteration, namely, group and zone rebalance, simple group rebalance or no rebalance at all.

5.2.2 Other printed output

A variety of other quantities that are descriptive of the case can be printed out. The presence or absence of these items from the case output is determined by the user's input, as described in Section 4.3. Heading information is supplied with each item to make it self-descriptive.

6. CODE STRUCTURE

For a transport theory code to be capable of managing large problems, it was once necessary for the code to make provision for saving much of the information in its work areas on subsidiary storage, i.e. magnetic tape or disk. The ANISN code [Engle 1967], the best known of such codes, incorporated efficient and elegant methods for achieving this result. From the time when development of ANAUSN was started, it was reasonably clear that the available size of random access memory (RAM) would increase dramatically. ANAUSN was therefore designed on the basis that all information required for the calculation would be held in RAM. For a time this limited the size of problem which could be handled, but the parallel development of multi-programming operating systems with essentially unlimited virtual memory allocation has alleviated this problem.

When execution of ANAUSN begins, the MAIN routine starts an internal clock, determines the amount of RAM available and passes this to the subroutine CONTRL which supervises the calculations, returning to the main routine only at the completion of the task. CONTRL reads as much of the input data as is necessary to determine the problem size before computing address pointers for the different arrays necessary for solution of the problem. The remaining input data are then read into the appropriate array positions before the subroutine RUN is called. This and other subroutines called use variably dimensional FORTRAN arrays with dimensions set from the routine CONTRL.

The main task of subroutine RUN is to handle eigenvalue search problems if these have been requested. For each possible reactor configuration during the search, a call is made to the subroutine POWER to determine the multiplication factor k_{eff} of the configuration but for non-search problems, a single call to POWER is sufficient.

The POWER subroutine actually supervises the iterative solution of the multi-group transport equations. At the beginning of an iteration, a set of group scalar fluxes will be available as well as the Legendre polynomial weighted integrals of the group angular fluxes necessary for the solution of problems with anisotropic scattering. From the fluxes, the spatial dependence of the fission density is calculated, then for each group in turn a call is made to an INNER subroutine to determine the angular and scalar fluxes in each reactor interval for that group. From the results, the POWER subroutine can determine whether the solution has converged or whether further iterations are necessary. For thermal problems in which the group structure allows upscatters to occur, the POWER subroutine also applies zone and/or group rebalance techniques to accelerate the convergence. When appropriate, a standard Chebychev acceleration of the fission density is also applied. The rebalance/acceleration techniques are discussed in more detail in Section 7.

It was advantageous to provide two versions of the INNER subroutines. The normal version handles the solution of the transport equation in an individual group by techniques which are now standard - see for example the discussion by Carlson [1963] and the description of the Winfrith DSN code by Francescon [1963]. Implementation of these techniques in FORTRAN means that before computing an angular flux in any direction in any mesh interval, it is necessary to test whether anisotropic scattering applies and whether fixed sources are present. For an important class of problems - some reactor cell calculations for example - neither of these situations will apply and a special truncated INNER subroutine called FASTIN is accessed in which these tests need not be made. Acceleration of the inner iterations is carried out by means of zone rebalance.

7. REBALANCE PROCEDURES

Acceleration of both inner and outer iteration loops in ANAUSN is achieved by use of rebalance procedures which are generalisations of the scaling procedure first proposed by Carlson and Bell [1958]. The inner iteration rebalance is essentially that developed by Engle and Mynatt [1968] who

calculated and used a different rebalance factor for each interval. The ANAUSN modification requires the use of only one rebalance factor for each rebalance zone which typically will contain about four mesh intervals. This is less subject to instabilities.

For the outer iteration loop let us use the group and zone rebalance scheme alluded to previously. Suppose that we are performing the (n)-th outer iteration and have just converged the fluxes for group g. The source term for group g will have included inscatters from other groups and these inscatters will have been calculated using the most recent flux estimates in the groups. Some of these estimates will be those obtained during the present (n)-th iteration and others will be those obtained from the previous (n-1)-th iteration. If the volume integrated group g flux in zone i found during iteration n is denoted by ϕ^n_{gi} , these zone fluxes will satisfy the equation

$$Q_{g,i} + R^n_{g,i-1} + L^n_{g,i+1} + \sum_{g'<g} \sigma_{gg',i}\phi^n_{g',i} + \sum_{g'>g} \sigma_{gg',i}\phi^{n-1}_{g',i} = R^n_{g,i} + L^n_{g,i} + (\sigma_{g,i} - \sigma_{gg,i})\phi^n_{g,i} \qquad (1)$$

where

$\sigma_{gg',i}$ is the macroscopic scattering cross section in zone i for scattering from group g′ into group g;

$\sigma_{g,i}$ is the total cross section in zone i group g;

$Q_{g,i}$ is the total non-scattering source into zone i group g including fission sources if present;

$R^n_{g,i}$ is the flow of particles leaving zone i in group g travelling to the right; and

$L^n_{g,i}$ is the corresponding flow travelling to the left, both flows being calculated during the present iteration (n).

So $R^n_{g,i} - L^n_{g,i+1}$ is the net current to the right at the interface between zones i,i+1.

In Equation (1), the presence of the $\phi^{n-1}_{g,i}$ terms which are not identical with $\phi^n_{g,i}$ shows that the solution is not yet converged. The rebalance scheme assumes that if the $\phi^n_{g,i}$ terms were multiplied by appropriate scale factors $F_{g,i}$, these results would be the final correct solutions and would satisfy the equation

$$\begin{bmatrix} Q_{g,i} + R^n_{g,i-1}F_{g,i-1} + \sum_{g'<g} \sigma_{gg',i}\phi^n_{g',i}F_{g',i} \\ +L^n_{g,i+1}F_{g,i+1} + \sum_{g'>g} \sigma_{gg',i}\phi^n_{g',i}F_{g',i} \end{bmatrix} = \begin{bmatrix} L^n_{g,i}F_{g,i} + (\sigma_{g,i} - \sigma_{gg,i})\,\phi^n_{g,i}F_{g,i} \\ +R^n_{g,i}F_{g,i} \end{bmatrix} \qquad (2)$$

The particle current terms have been multiplied by the appropriate scale factors.

If Equation (1) is multiplied by $F_{g,i}$ and Equation (2) subtracted from the result, we recover the equation

$$\begin{bmatrix} -R^n_{g,i-1}F_{g,i-1} - L^n_{g,i+1}F_{g,i+1} - \sum_{g'\neq g} \sigma_{gg',i}\phi^n_{g',i}F_{g',i} \\ +\left(\sum_{g'<g} \sigma_{gg',i}\phi^n_{g',i} + \sum_{g'>g} \sigma_{gg',i}\phi^{n-1}_{g',i} + Q_{g,i} + R^n_{g,i-1} + L^n_{g,i+1}\right)F_{g,i} \end{bmatrix} = Q_{g,i} \qquad (3)$$

There remains the question of what to do with Equations (3) for the first zone i=1 and the last zone. An examination of the physical situation described by Equation (1) for the end zones shows that the terms $R^n_{g,0}$ and $L^n_{g,NZ+1}$ should simply be removed from Equations (3). When this is done, all Equations (3) can be cast into a set of block tridiagonal matrix equations which can be solved by block forward elimination and back substitution. Should there be only one rebalance zone, Equations (3) reduce to those for the global group rebalance scheme initially proposed for the acceleration of upscatter problems by Clancy and Donnelly [1970].

As formulated, Equations (3) lead to separate rebalance scale factors for each zone and each group. For those groups unaffected by upscattering processes, the scale factors will be unity if the inner iteration process for these groups has converged. The rebalance scale factors need only be determined for those groups with upscatters and this is implemented within ANAUSN.

The ANAUSN outer iteration acceleration scheme is to apply the global rebalance scheme almost always, replacing it periodically with use of the group and zone rebalance factors which are solutions of Equations (3). The frequency and maximum number of applications of the group and zone rebalance is determined by input parameters, as described in Section 4.3. As a particular case gets close to convergence, the group-zone rebalance factors tend to unity. When they are sufficiently close, the group-zone rebalance is disenabled and global rebalance used on all outer iterations. At this stage of the solution, the extra computational effort needed to determine separate rebalance factors for each zone and group seems not to be worthwhile.

After zone-group rebalance has been disenabled some problems with fission may still be slow to converge. For this reason, a simple Chebychev acceleration of the fission density is initiated.

8. INVOCATION OF ANAUSN

As a module within the AUS scheme the code can be invoked by the AUSYS statement

```
link anausn
```

which is an abbreviation of the standard linkage statement

```
link anausn(1,dd2),(2,dd12),(8,dd21),(9,xs1),(10,gm1),
(11,fl2),(12,dd22),(22,xs2),(23,xs3)
```

If data for ANAUSN are not in dd2 but in another dataset dd*n*, the appropriate AUSYS statement is

```
link anausn(1,ddn)
```

9. REFERENCES

Bennett, N.W., Pollard, J.P. [1967] - SCAN - A Free Input Subroutine for the IBM360. AAEC/TM399.

Carlson, B.G. [1963] - The Numerical Theory of Neutron Transport. In Methods of Computational Physics (Alder, B., Fernbach, S., Rotenberg, M. Eds.). Vol.1, Academic Press, New York, p1.

Carlson, B., Bell, G. [1958] - Solution of the Transport Equation by the S_N Method. Proc. 2nd UN Conf. on the Peaceful Uses of Atomic Energy, Geneva, 16:535.

Carlson, B., Lee, C., Worlton, J. [1960] - The DSN and TDC Neutron Transport Codes. LAMS-2346.

Clancy, B.E. [1982] - ANAUSN - A One-Dimensional Multigroup SN Transport Theory Module for the AUS Reactor Neutronics System. AAEC/E539.

Clancy, B.E., Donnelly, I.J. [1970] - Outer Iteration Scaling in Neutron Transport Codes. Nucl. Sci. Eng., 39:398.

Engle, W.W.Jr. [1967] - A Users' Manual for ANISN. Union Carbide Corporation, Nuclear Division. K-1693.

Engle, W.W.Jr., Mynatt, F.R. [1968] - A Comparison of Two Methods of Inner Iteration Convergence Acceleration in Discrete Ordinate Transport Codes. Trans. Am. Nucl. Soc., 11:193.

Francescon, S. [1963] - The Winfrith WDSN Program. AAEW-R273.

ICPP — COLLISION PROBABILITY MODULE

1. INTRODUCTION

The isotropic collision probability program (ICPP) is a general purpose collision probability module for one-dimensional geometries and for the rod clusters typical of pressure tube reactors. It may be used within the AUS system either for few-region cell calculations suitable for group condensation or for many-region calculations after group collapsing. The collision probability approach is most suited to few-region calculations but also is preferable to S_N methods for detailed calculations in slab cells and clusters.

The ICPP module calculates first flight collision probabilities with the usual assumptions that neutron sources, including those due to scattering, are isotropic and spatially flat within each mesh interval. The module also solves the multigroup collision probability equations for eigenvalue or fixed source problems. Where anisotropy of scattering is important, an alternative such as the S_N method should be used instead. Where the source gradient is large, ICPP requires many more mesh intervals than a corresponding S_N calculation.

The module includes a variety of collision probability routines which vary from very fast approximate routines through normal routines to slow routines most suited to few-group check calculations. Details of the methods used in the various routines and the accuracy obtained have been given by Doherty [1] [2] [3] and Robinson [4]. These reports include intercomparisons of the various routines for a number of simple cells and standard clusters. The method of solution for the multigroup collision probability equations has been given by Doherty [5]. This report is restricted to a brief description of the various routines and a description of input to ICPP.

2. BRIEF DESCRIPTION OF COLLISION PROBABILITY ROUTINES

A selection from the set of collision probability routines in ICPP may be made indirectly as described in Section 3, or by referring to a particular routine by number. Each of the routines has been given a name of the form PROBn where n is an integer associated with each routine.

2.1. PROB1 – Free Slab Routine

In this routine, a boundary condition of zero incoming angular flux is imposed on both left and right boundaries. Collision probabilities are simple E_3 exponential functions so the routine is very fast. With only limited application of collision probabilities to free boundary problems, a routine with reflected and free boundaries has not been included. Void regions must be omitted when using this routine. Further details are given by Doherty [1].

2.2. PROB2 – Reflected Slab Routine

A reflective boundary condition is imposed at both left and right boundaries. Collision probabilities are evaluated by Gaussian quadrature in μ, the cosine of the angle between the neutron direction and the direction normal to the slab. Void regions must be omitted. Further details are given by Doherty [1].

2.3. PROB3 – Periodic Slab Routine

The system of slabs repeats from left to right and is taken to be infinite. Again, the collision probabilities are obtained by Gaussian quadrature in μ, and void regions must be omitted. Further details are given by Doherty [1].

2.4. PROB4 – Free Spherical Routine

In evaluating the collision probabilities, only neutron paths normal to a reference radial direction need be considered. The integration over r within each mesh interval is performed using Gaussian quadrature. A boundary condition of zero incoming angular flux is imposed on the surface. Further details are given by Doherty [2].

2.5. PROB5 – White Reflected Spherical Routine

This routine uses the collision probabilities evaluated for a free boundary condition on the sphere (PROB4) together with a surface/volume reciprocity relation to include the white reflective boundary condition. The routine is intended to be used for an array of spheres with a constant spacing or for dispersed fuel particles.

2.6. PROB6 – Free Cylinder Routine Using Numerical Integration

In this routine, the probabilities are evaluated by considering the projection of the neutron paths in the x-y plane. Only paths normal to a reference radial direction need be considered. The integration over the z direction cosine is performed analytically using K_{i3} Bickley functions. The integration over r within each mesh interval is performed by Gaussian quadrature. Further details are given by Doherty [1].

2.7. PROB7 – White Reflected Cylinder Routine (Numeric)

This routine uses the collision probabilities evaluated for a free boundary condition on the cylinder (PROB6) together with a surface/volume reciprocity relation to include the white reflective boundary condition.

2.8. PROB8 and PROB9 – Approximate Cylinder Routines

The PROB8 routine is used for cylinders with a free boundary condition and the PROB9 routine for cylinders with a reflective boundary. These routines use an approximate method given by Robinson [4] and based on the work of Bonalumi [6] [7].

The method allows the outer boundary to be either circular or polygonal. A white reflective boundary condition may be applied on an outer circular boundary. For a reflective polygonal boundary, the boundary condition is restricted to a two-region system, which is formed by volume smearing the actual geometry as specified by the user. Normally, the inner region is fuel and the outer region is smeared cladding and coolant. These routines are very fast and suitable for few-region many-group calculations to provide condensation spectra.

2.9. PROB10 – Cylinder With Mirror Reflection

Double Gaussian integration is required in this routine. One integration is over the r mesh, as in the PROB6 routine, the other is over the z direction cosine, which can no longer be performed analytically. The routine is slow and used only for check calculations. Further details are given by Doherty [1].

2.10. PROB11 – Cylinder With Square Reflective Boundary

This routine uses a double Gaussian integration. As in the PROB6 routine, projections of neutron paths normal to a radial direction are considered. One integration is over the r mesh, with the integration over the z direction cosine being in terms of the K_{i3} function. The second integration is over the angle between the radial direction and a reference side of the square. The routine is slow and used only for check calculations. Further details are given by Doherty [1].

2.11. PROB12 – Cylinder With Hexagonal Reflective Boundary

The routine is similar to the PROB11 routine, but a non-orthogonal co-ordinate set is used to relocate neutrons after reflection. The routine is slow and used only for check calculations.

2.12. PROB13 – Rod Cluster Numerical Routine

This routine integrates numerically over the entire cluster cell which is taken to have a white reflecting outer boundary condition. The integration is performed by laying out a set of parallel lines in the x-y plane at a number of discrete angular orientations to a reference diameter of the cluster. The integration along each line is performed using K_{i3} functions for the integration over the z direction cosine. This routine is slow and used for few-group calculations. Further details are given by Doherty [3].

2.13. PROB15 – Rod Cluster Semi-Numerical Routine

This routine is similar to PROB13 but the numerical integration is applied only to the inner annuli of the cluster. An approximate method of the Bonalumi type is applied to the outer annuli. The numerical integration is used for one more annulus than the number which contain rods. This routine is slow and used for few-group calculations. Further details are given by Doherty [3].

2.14. PROB19 – Rod Cluster Approximate Routine

This is an approximate routine for clusters with a white reflective or free outer boundary condition. The basis of the calculation is the division of the cluster into pincells which each contain a fuel rod and a portion of the coolant. Reflective boundary conditions on these pincells form part of the solution. The collision probabilities are formed by a synthesis of collision probabilities for an average pincell in each ring of rods and collision probabilities for the entire cell with smeared cross sections. The pincell and smeared-cell collision probabilities are calculated by the method used in PROB8 and PROB9. This routine is fast and suitable for most applications. Further details are given by Robinson [4].

2.15. PROB108 and PROB109 – Approximate Cylinder Routines

These routines are another pair of approximate routines for a cylinder and are similar to the pair PROB8 and PROB9 but are based on an earlier model of Bonalumi [6]. The PROB8 and PROB9 routines are preferred because they provide for polygonal as well as circular boundaries. PROB108 is used for a free boundary and PROB109 for a circular white reflecting boundary. The method of these routines is used for the outer annuli in PROB15. Further details are given by Doherty [1].

3. INPUT DESCRIPTION

3.1. General

Input data on Fortran unit 1 are in free format and are read with the SCAN input routine [8]. The input is in the form of keywords, which are given in **bold** in the following description, followed as necessary by an appropriate string of numeric data. Items that the user will replace with an actual value are given in *italics*. More than one keyword may be given on each input record from which only the first 72 characters are processed. The data are supplied in blocks and all data required for one block should be supplied before any data are given for a succeeding block.

The input routine first attempts to read from three AUS data pools: geometry data from Fortran unit 13; cross-section data from Fortran unit 14; and a FLUXB data pool from Fortran unit 11. Any data obtained from these data pools are modified if the appropriate input data are supplied.

The FLUXB data pool may be used as a flux guess. Output AUS data pools are written on the same Fortran units as those used for input.

Default values for all input data items are available. Therefore, no input data at all (*i.e.* an empty data set) may be given if geometry and cross-section data pools are available and the standard options are suitable.

3.2. Input Block 1

This block consists simply of a single line which is used as a title. It must be given if any other input data are supplied.

3.3. Input Block 2

The user may enter any of the following data.

np(i)=n

or

np(i-j)=n where i, j, n are integers with $0 < i < j \leq 36$ and $n \geq 0$, is used to alter default values of the triggers **np**(1) to **np**(36) which control printing. The two options change a single trigger or a range of triggers. Printing is suppressed for trigger values greater than zero. The trigger **np**(i) refers to cross sections (i=2), fission spectra (i=3), fixed source (i=4), geometry (i=6), flux guess (i=7), calculated fluxes (i=35), collision probabilities (i=36). The default is **np**(1-36)=1.

eps followed by a real number is used to specify the tolerance on the eigenvalue, or the activations (normally absorptions) in a source calculation. The default is .0001.

ngauss followed by an integer specifies the order of Gaussian integration (default 16). The default is the maximum order available.

lines followed by an integer specifies the number of lines per annulus for PROB13 or PROB15.

nangles followed by an integer specifies the number of angles for PROB13 or PROB15. The integration is over the range 0 to 2π and the number of angles should be neither a divisor of, nor be divisible by, the number of rods on any ring of the cluster.

3.4. Input Block 3

The data supplied in this block either specify the complete geometry of the system or are used to modify the data obtained from the AUS geometry data pool.

jom followed by an integer specifies geometry type. The integer has the values 0 for plane, 1 for cylinder and 2 for sphere. The default is 1.

ibr followed by an integer specifies the right hand boundary condition. The integer has the values 0 for free, 1 for mirror reflective, 2 for periodic and 3 for white reflective. The default is 3 for curved boundaries, otherwise 1.

square

and

hexagon specify the shape of the outer boundary in cylindrical geometry if numerical routines are used. The shape is circular by default.

bound followed by a real and an integer specifies the boundary for the approximate cylinder routine or the boundary between rods in the approximate cluster routine. The real (normally an integer) specifies the number of sides of a polygonal boundary or is zero for a circular boundary. The integer specifies the number of inner annuli to be included in the first region of a two-region system constructed in applying a reflective boundary condition to a polygon. The integer is zero for a circular white reflective boundary or is negative for a free boundary. In the cluster routine, the specification of a free boundary applies to the outer boundary of the cluster whereas other conditions apply to the boundary between rods. The default is a circular white reflecting boundary.

bonalumi
and
numerical specify whether approximate or numerical methods are to be used for cylinders and clusters. The default is **bonalumi**.

iprob followed by an integer specifies the collision probability routine directly by number rather than indirectly with the preceding entries.

ndiv followed by an integer specifies whether, in the PROB13 routine, coolant annuli are used to subdivide the rods. The default is no such subdivision. The integer gives the number of radial subdivisions of a rod which are to be further subdivided by coolant annuli.

ni followed by an integer, *ni*, specifies the number of mesh intervals. This entry should be followed by a mesh interval specification:

r followed by (*ni*+1) reals giving the mesh interval positions; or

dr followed by *ni* reals giving the mesh interval spacings. r_1=0 is implied when using **dr** and must be specified when using **r**. ($r_{i+1} = r_i + dr_i$)

mni followed by *ni* integers specifies the material numbers for each mesh interval, or the materials may be specified by zones as in the ANAUSN module using the three entries:

nz followed by an integer, *nz*, to specify the number of zones;

mzi followed by *ni* integers giving the zone number for each interval; and

mnz followed by *nz* integers giving the material number for each zone.

The following two entries are used to specify the rods in cluster geometry. The rods are taken to be arranged on a number of rings.

rodsub *n*, *j*, (dr_i, *i*=1,*j*) , (mat_i, *i*=1,*j*)
where *n* is the number of the ring, numbered from the centre outward,
j is the number of radial subdivisions of a rod in this ring,
dr_i specifies the mesh interval spacings of the subdivision, and
mat_i specifies the material number in each subdivision.

array *n*, *nrod*, *prod*, *qrod*
where *n* is as for **rodsub**,
nrod is the number of rods in this ring,
prod is the radius of the circle on which the rod centres lie,
qrod is the angular displacement of one rod from a reference diameter of the cluster. The rods are taken to be equally spaced around the ring. The parameter *qrod* may be given in radians or degrees.

3.5. Input Block 4

Data in this block consist of directives which stop the writing of AUS data pools. The default option is to write the data pools.This may be altered using any of

nowgm

nowxs

nowfl

for geometry, cross-section and flux data pools respectively.

3.6. Input Block 5

This block is given only if all cross sections are to be entered in the input stream. Partial modification of the data from the AUS data pool is not supported. The cross sections are entered as one block following a single keyword.

ng ng, ((lps_{in}, lv_{in}, act_{in}, nuf_{in}, (grp_{jin}, j=1, lv_{in}), i=1, ng), n=1, $nmat$), (χ_i, i=1, ng)
where ng is the number of groups,
lps is the position of the self-scatter cross section in the vector grp,
lv is the length of the vector grp,
act is the activation cross section, normally absorption,
nuf is the fission emission cross section,
grp is a vector giving the absorption cross section followed by the outscatters from the group,
$nmat$ is the largest material number referred to in the geometry description,
χ is the fission spectrum which is used for all materials. (A material dependent spectrum is supported when cross sections are obtained from an AUS data pool.) A single zero value should be entered if a fission spectrum is not needed.

3.7. Input Block 6

The code performs an eigenvalue (k_{eff}) calculation unless it is directed to perform a calculation with a fixed volume source. The source calculation is performed if a source is entered using

qv ngs, ((qv_{ij}, i=1, $nreg$), j=1, ngs)
where ngs is the number of groups with a fixed source,
qv_{ij} is the source density in interval i for group j,
$nreg$ is the number of spatial intervals.

The end of the input data is signalled by including the keyword

end

Only one case may be done each time the module is executed.

4. REFERENCES

[1]Doherty,G. [1969] - Some methods of calculating first flight collision probabilities in slab and cylindrical lattices. AAEC/TM489.

[2]Doherty,G. [1969] - Solution of some problems by collision probability methods. AAEC/E199.

[3]Doherty,G. [1970] - Collision probability calculations in cluster geometry. AAEC/E205.

[4]Robinson,G.S. [1979] - Approximate first collision probabilities for neutrons in cylindrical and cluster lattices. AAEC/E465.

[5]Doherty,G. [1969] - Solution of the multigroup collision probability equations. AAEC/E197.

[6]Bonalumi,R.A. [1961] - Neutron first collision probabilities in reactor physics. Energ. Nucl. (Milan), **8**:326.

[7]Bonalumi,R.A. [1965] - Systematic approximations in neutron first collision probabilities. Energ. Nucl. (Milan), **12**:1.

[8]Bennett,N.W. & Pollard,J.P. [1967] - SCAN - a free input subroutine for the IBM360. AAEC/TM399.

EDITAR — REACTION RATE EDITING AND CROSS-SECTION AVERAGING MODULE

1. INTRODUCTION

The AUS neutronics code system has been developed at Lucas Heights for application to a wide range of neutronics calculations including thermal reactors, fast reactors and fusion blankets. The EDITAR module of AUS is a general purpose module for the editing of fluxes, which may be used either for lattice cell calculations or global calculations. An early version of EDITAR was originally described by Robinson [1].

EDITAR may be used

- to print reaction rates for both the materials in the system and their constituent nuclides;
- to average cross sections over space and energy, and either print the results or produce an AUS cross-section data pool;
- to apply bilinear weighting in forming reaction rates and thus obtain material worths;
- to apply bilinear weighting in cross-section averaging;
- to distinguish between neutron and photon groups in order to print reaction rates separately for neutrons and photons;
- to calculate the leakage from a cell using a B_L calculation; and
- to "unsmear" the fluxes in one EDITAR calculation by using the smearing factors calculated in a previous EDITAR calculation.

The module may be used, for instance, to obtain detailed reaction rates of nuclides within a cell from a global flux calculation which uses cell-average cross sections.

EDITAR may be applied to either one- or two-dimensional calculations. The input flux file may be either an AUS data pool or the VARFLM flux file produced by DORT [2]. As most editing is done by material, the number of dimensions has little effect. An option is available to print reaction rates at each mesh interval. Under this option, all reactions are printed at each mesh interval for 1D cases but only one reaction, *e. g.* fission, is printed for 2D cases.

2. USE OF AUS DATA POOLS

The EDITAR module normally obtains cross-section input for materials and nuclides from the AUS data pool on Fortran unit 10. These cross sections may be supplemented from a second AUS cross-section data pool on Fortran unit 11. The output AUS data pool for cross sections is normally written on unit 12. Where a second set of input cross sections is used, it is intended that the first set of data should be for materials, whereas the second set should be for the nuclides present in those materials. The latter set of data may be in a different group structure from the first, if the first group structure is a condensation of the second. (Condensation is the process of reducing the number of groups by combining some groups and calculating average cross sections for groups in the new set.) The purpose of this option is to enable a sequence of calculations in which a nuclide cross-section data pool remains uncondensed while the material cross sections are condensed and spatially smeared, possibly through more than one stage.

The geometry is obtained from the geometry data pool on Fortran unit 7. The input flux is obtained from the flux data pool on Fortran unit 9, which may be either of the FLUXA or FLUXB type. If a FLUXA type, which specifies mesh-edge fluxes, is given these are first converted to mesh-average fluxes. This conversion has no effect on reaction rates, but does affect quantities related to leakage between material zones. A DORT VARFLM file may be substituted, but this must have only one I-mesh and only the scalar flux is used.

The data on material composition and possible material smearing are obtained from the pair of STATUS data pools **st1** and **st2**. The **st1** data are taken from Fortran unit 4 and the **st2** data from unit 25. EDITAR may be used without these data, in which case only the materials directly in the geometry specification are considered to be in the system.

For cell calculations, the latest cell name is taken from **st2** and only material definitions for that cell name are read from **st1**. When a B_L calculation is performed, a **gfac** entry giving the ratio of k_{eff} to k_∞ flux in each group is added to both **st1** and **st2**. When AUS cross sections are written for any material, a **grps** entry followed by a mixing rule entry for each output material are added to **st1**. The mixing rule entry is made even if the material is simply transferred from the input to the output cross-section data pool. Normally, no entries are made for any output nuclides and their names remain unchanged. The 20-character name of output materials is constructed to maintain uniqueness of material names.

3. CALCULATION OF LEAKAGE FROM A CELL

EDITAR includes a B_L flux calculation which may be applied to the calculation of leakage from a lattice cell. To obtain a homogeneous system to which the B_L equations can be applied, cell-average cross sections are formed by flux weighting using the input flux. Simple flux weighting is used even if input spatial currents are available. A transport corrected P_0 calculation is performed rather than a B_0 calculation. For P_0 calculations, the input transport cross sections are flux weighted before forming the cell average diffusion coefficient. EDITAR does not include a streaming treatment.

The solution of the B_L equations is exactly the same as that in the MIRANDA module from which the flux solution [3] subroutine used in EDITAR was taken. When bilinear weighting is requested, the adjoint B_L equations are also solved.

A flux within the cell which includes leakage effects is obtained by multiplying the homogeneous flux from the B_L calculation by the ratio of input spatial flux to the integrated input flux. The same ratio is used to multiply any homogeneous currents calculated from the B_L equations. Even if spatial currents are available, they are ignored in cell calculations unless the user directs otherwise (see Section 5.2). In that case, any leakage currents are ignored. Input adjoint distributions are always ignored when a cell leakage calculation is performed.

4. FORMULAE USED IN AVERAGING

These formulae are, with two exceptions, unchanged from those given in the original report [1].

In averaging the transport cross section using flux weighting in a global calculation (Equation (4) [1]), D_{ig} is the diffusion coefficient of the material at interval i except when the cross section being formed is a mixture of materials actually in the geometry (spatial smearing). In that case, D_{ig} is the diffusion coefficient of the mixture before group collapsing.

In averaging the transport cross section using bilinear weighting in a global calculation (Equation (29) [1]), the term A_{ig} is replaced by its absolute value.

5. INPUT DESCRIPTION

5.1. General Layout

Input data to EDITAR are in free format and are read with the SCAN input routine [4]. The data are given in the form of a number of entries which each consist of a keyword followed by a string of numeric or alphabetic data items. More than one entry may be given on each input record from which only the first 72 characters are processed.

Most of the input parameters have been given default values. Thus, if the standard value is required, an entire entry may be omitted or trailing data may left off the end of a data string. The data entries are grouped into three sets. The first set gives the major parameters which establish the type of calculation to be performed. This set must be given first because they are processed before the AUS data pools are read. The second set provides parameters for the leakage flux calculation if this is required. The third set specifies the output requirements. The default calculation is the generation of an AUS output file with one-group cross sections for each material included in the geometry.

The entries of each type are described in the following subsections. The following conventions are used.

- Information to be reproduced exactly is given in **bold**.
- Items that the user will replace with an actual value are given in *italics*.
- Optional items are given inside [ ].
- Examples and defaults are given in `constant width` type.

5.2. Major Parameters

5.2.1. use cross-section data specification

use *iuxs*1 *iuxs*2

where *iuxs*1 is the Fortran unit number of the main input cross-section data pool (default 10), and
*iuxs*2 is the Fortran unit number of an additional input cross-section data pool giving the isotope data (default 0).

The number of groups on *iuxs*1 and *iuxs*2 may be different, with the *iuxs*1 group structure being condensed from the *iuxs*2 structure, except when bilinear weighting is used.

5.2.2. calc calculation type

calc [**cell**] [**multicell**] [**global**] [**reactor**]

where only one of the four types should be given.

cell indicates a calculation in which only one set of data is required from the STATUS data pool and a cell diffusion coefficient is used in weighting transport cross sections.

multicell is similar to **cell** but all data are required from the STATUS data pool.

global or **reactor** indicates a complete calculation as opposed to a **cell** calculation. Current weighting of transport related cross sections is used if currents are available on the flux data pool.

The default calculation type, which is **global** if the geometry includes a free boundary and **cell** if there is no free boundary, should be satisfactory for most calculations.

5.2.3. weight cross-section weighting

weight [**flux**] [**bilinear**] [**current**]

where only one of the three weights should be given,
flux causes flux weighting to be used,
bilinear causes bilinear, *i.e.* the product of flux and adjoint, weighting to be used, and
current causes current weighting to be used for transport related processes if input currents are available.

This specification over-rides the **flux** or **current** weighting which is implicit for each calculation type.

5.2.4. scalef flux scaling

scalef *flxf*

where *flxf* is a factor by which the input fluxes are to be multiplied.

This parameter provides a simple means of renormalising the input flux which, in eigenvalue problems, would otherwise be normalised to one neutron emitted in fission.

5.2.5. print control of printed output

print *i*

where *i*=-1 for a minimal print,
i=0 for a normal print (the default),
i=1 for additional print including nuclide names, mixing and smearing table,
i=2 for debug prints.

Though not a major parameter **print** may be given with this set of entries to print the information from the input data pools.

5.3. Flux Calculation Parameters

This set of parameters controls the leakage flux calculation which is applied by default to calculations with no free boundary condition. Most of the entries are the same as the MIRANDA module.

5.3.1. kinf no leakage calculation

kinf

where this keyword is given if no leakage calculation is required.

5.3.2. buck group buckling

buck [**b***n*] $bgsq_1$ $bgsq_2$ ...

where **b***n* specifies the order of the B_L flux solution and is optional,
$bgsq_i$ gives the buckling for energy group *i*.

The default for **b***n* is the order of the input cross sections. If a short list of $bgsq_i$ is given, the last value in the list is used for the remaining groups. Thus one value is sufficient for a constant group buckling. The default buckling is 0.001 cm^{-2}. A transport corrected P_0 rather then B_0 calculation is performed.

5.3.3. search search for critical buckling

search on bsq for *k dk*

or

search off

where **on, off** turns the search on or off and the default is **on**,
k is the value of k_{eff} required – default 1.,
dk is the accuracy to which *k* is required – default 0.0002.

5.3.4. fluxparm flux options seldom required

fluxparm *limfl limsea nscale acc*

where *limfl* is the limit on the iterations to converge the flux calculation – default 100,
limsea is not used,
nscale is the number of broad groups used in scaling the thermal flux to improve convergence – default 3, and
acc is the required accuracy – default 0.0001.

5.4. Output Requirements

5.4.1. output type of output required

output [**rr**] [**xs**] [**aus**] [**p***n*] [*iout*] [**scan** [*irn*]] [**punch**]

where a selection of the parameters should be made,
rr specifies printed group reaction rate output,
xs specifies printed group cross-section output,
aus specifies AUS cross-section data and STATUS data output,
p*n* is the scattering order for AUS output – default is the same order as for input cross sections,
iout is the Fortran unit number for AUS cross-section output – default 12,
scan specifies printed one-group microscopic reaction rate output at each mesh interval,
irn is the AUS reaction number for which the **scan** is given,
punch causes the reaction rates from **scan** *irn* to be written on Fortran unit 2.

The *irn* value is meaningful only for 2D geometry or **punch** and has a default value of 6 (fission). A value for *irn* must only be entered immediately after **scan**.

The *irn* parameter takes the values

1 to 8	the corresponding neutron reaction rate on an AUS library,
9	neutron kerma,
14	photon scattering,
17	photon photo-electric reaction,
18	photon pair production,
19	photon kerma,
29	total kerma (neutron plus photon).

The default output type is **aus**. The default for **rr** is macroscopic reaction rates which include the concentrations of nuclides and the region volume for both nuclides and materials. The **rr** output with bilinear weighting gives material worths and also fission and capture flux-weighted reaction rates. The output from **scan** is flux weighted only.

5.4.2. groups condensed group structure

groups n iu il_1 il_2 ... il_n

where n is the number of condensed groups,
iu is the first input group of condensed group 1,
il_i is the last input group of condensed group i.

The default is one group. Coupled neutron/photon cross sections are imagined to form a single set with neutrons first. A condensed group should not include both neutron and photon groups.

5.4.3. **where** mesh interval labelling

Mesh interval labels are used to denote a set of mesh intervals over which a material may be averaged. EDITAR generates a label for each material directly specified in the geometry, which labels the intervals containing a material with the name of that material. Thus if a 5-interval calculation had a material named `fuel` in intervals 1 and 2, and a material named `cool` in intervals 3, 4 and 5, then the interval set (1, 2) is given the label `fuel` and the interval set (3, 4, 5) is given the label `cool`. The label `all` is also automatically given to the complete set of intervals. Additional interval labelling may be established using a number of **where** entries which should all be given before the first **start** keyword.

where *name* j_1 j_2 ... j_n

or

where *name* wh_1 wh_2 ... wh_n

where *name* is the label to be given to the set of n intervals,
j_i is the i^{th} interval of the set, and
wh_i is a **where** label which has been previously defined by the user or generated by EDITAR.

In 2D geometry the intervals are imagined to form a single vector when **where** is used. The input becomes easier if the **where2d** entry is used.

where2d *name* j_1,k_1 j_2,k_2 ... j_n,k_n

where *name* is the label to be given to the set of n intervals,
j_i is the x or r mesh interval number of the i^{th} mesh box, and
k_i is the y or z mesh interval number of the i^{th} mesh box.

Alternatively, both j_i and k_i may specify a range of intervals. The range is indicated by a pair of numbers in which the second is negative. If a range is given for either j_i or k_i, it must also be given for the other direction. For example,

```
where2d 2-5,6-8 2-5,11-11
```

is correct, but

```
where2d 2-5,6-8 2-5,11
```

is not.

5.4.4. **mixture** mixing of nuclide data

Mixing of materials in general is not allowed for in EDITAR. However, the **where** facility allows the mixing of materials explicitly in the geometry. Also, a limited mixing of nuclides is permitted using

mixture *name* nuc_1 nuc_2 ... nuc_n

where *name* is a label for the mixture, and
nuc_i is the name of a nuclide.

The limitation on this option is that a **mixture** may only be used in printing reaction rates, not in producing cross sections. The sum of the reaction rates in the named nuclides is printed. The option applies only to nuclides in the system, the appropriate volumes and concentrations being used in forming the result.

5.4.5. **reqd** output material selection

reqd $name_1$ [@lab_1] $name_2$ [@lab_2] ...

where $name_i$ is :

an input material or nuclide name, or
a **where** interval label, or
a **mixture** label, or
the name `cell` for a material which is the cell average, or
the name `allmat1` which implies all materials and nuclides on the first input cross-section file, or
the name `allmat2` which implies all nuclides on the second input cross-section file;

@ is the character @ to indicate an optional lab_i is given,
lab_i is a **where** interval label over which the $name_i$ material or nuclide is to be averaged.

The output materials are averaged over the appropriate regions using volumes and scalar energy-dependent fluxes as typical weights. The weighting actually used depends on the particular cross section, the type of **output**, the boundary condition on the AUS GEOM data pool, and the entries **calc**, **weight**, **micro**, and **macro**.

If a **reqd** entry is not given the default is `allmat1`. If @lab(i) is not given, $name_i$ is averaged over the intervals in which it is present, or over `@all` when it is not in any intervals. If $name_i$ is a **where** interval label rather than a material, a material is formed by averaging the materials in each of the intervals specified by the **where**. Thus the name `all` is equivalent to `cell`. The material `cell` is renamed with the name of the cell. The 20-character name of output materials is constructed to maintain uniqueness of material names. The name of an output nuclide remains unchanged unless a **where** interval is specified for the nuclide.

5.4.6. **average** averaging of nuclide data

More than one set of data for a nuclide may be included as input to EDITAR. The default option is to treat each set of data as if it were a distinct nuclide. If average output for some nuclides is desired this is obtained with the entry

average $name_1$ $name_2$ ...

where $name_i$ is the name of a nuclide for which the output data is to be averaged to form a single set of data for that nuclide. If the entry **average** is given without a following nuclide list, all nuclides will be averaged. An **average** entry must not be given before the **reqd** entry to which it applies.

5.4.7. **micro** and **macro** weighting of nuclide data

The default option (**macro**) is to print macroscopic reaction rates for those nuclides which are a constituent of geometry materials. The concentrations are obtained from the STATUS data pool. All macroscopic reaction rates, whether for nuclides or materials, include the volume of the region in which the nuclide or material is present. The output cross sections of a nuclide are microscopic but the concentrations are used in spatial averaging. To obtain microscopic reaction rates and to ignore concentrations in spatial averaging, the keyword required is

micro

Microscopic reaction rates are divided by the region volume. The keyword

macro

is used to restore the standard state.

5.4.8. scanwt group weights used with a reaction **scan**

scanwt gw_1 gw_2 ... gw_n

where gw_i is the weight to be given to input group i in a reaction rate **scan**, and
n is the number of input groups.

The default weights are all unity.

5.4.9. transport transport cross-section option

Transport cross sections following a B_L calculation are normally adjusted to give the correct leakage in a diffusion theory calculation of a reactor. If this is not appropriate, the entry required is

transport p1

The most likely reason for requiring this option is that the cross sections are to be used in a global S_N calculation with P_0 scattering.

5.4.10. Factors for dose calculations

In calculations of dose to tissue, it is necessary to multiply kerma data by a Relative Biological Effectiveness (RBE) factor. As there is no provision for including this factor on an AUS cross-section library, the factor may be entered here. To form total doses, a summation mechanism has also been included. The use of these factors is restricted to **rr** and **scan** type output only. The entry **qfactor** can only be used in conjunction with the **cfactor** entry.

cfactor c_1 c_2 ... c_n

qfactor q_1 q_2 ... q_n

where n is the number of "materials" on the **reqd** entry,
$|c_i|$ is a concentration factor used to multiply all reactions of **reqd** material i,
q_i is an RBE factor used to multiply neutron kerma data for **reqd** material i, and
if c_i is negative, the data for material i are added to those of the first previous **reqd** material j with positive c_j.

For example,

```
reqd tissue tiss b10  cfactor 2*1. -1.e-5  qfactor 2*1.6  2.0
```

would produce output for `tissue` and a material (labelled `b10` by EDITAR) giving the sum of `tiss` and 1.E-5 of `b10` reactions. Neutron kerma would be multiplied by 1.6 for `tissue` and `tiss`, and 2.0 for `b10`.

5.4.11. smear smearing factor options

When it is necessary to smear materials in one calculation for use in a following calculation, and reaction rate editing by nuclide is required in the second calculation, the normal procedure within AUS is to not condense the nuclide data and to "unsmear" into group fluxes in the original group structure. (Note that unsmearing of the adjoint flux is not supported.) If the nuclide data are being group-condensed, smearing factors in the condensed groups are required on the STATUS data pool. Alternatively, the STATUS data pool may be ignored completely and the volume fraction, smearing factors and the concentration must then be included in the nuclide cross section.

In performing reactor burnup calculations with microscopic nuclide cross sections in AUS using the MICBURN and CHAR modules, nuclide cross sections which include smearing factors are required. An EDITAR option is required to prepare this data.

To use any of these alternative treatments, the entry required in the EDITAR calculation which performs the smearing is

smear *jsm ksm*

where *jsm*=1 if the smearing factors are to be written in the condensed group structure,
jsm=-1 if no smearing factors are to be written,
ksm=1 if everything is to be included in the nuclide cross section,
ksm=-1 if smearing factors are to be included in the nuclide cross section.

The default values are zero for both parameters. A value of $ksm \neq 0$ may be used when the **cell** material or nuclides are being produced, but not otherwise. The *ksm*=1 option is meant to be used when generating nuclide data for reaction rate editing by a module which does not use the STATUS pool. The *ksm*=-1 option is meant for use with MICBURN. The *jsm*=-1 option also is meant for use with MICBURN. Additionally, the options $ksm \neq 0$ or *jsm*=-1 result in names of materials and nuclides not being modified, even if they are not in the system or a **where** interval label is given.

5.4.12. **gamsource** output gamma source

It is sometimes convenient to produce a gamma source from one calculation for use in another. To assist with this procedure, provision has been made for printing the gamma source. The keyword

gamsource

causes output of the photon production term for the regions specified on the **reqd** entry when **rr** output is requested. The group structure is that of the flux data pool.

5.4.13. **start** initiate calculation

The keyword

start

is required to initiate the calculation set up by the preceding input. More than one **start** keyword may be given. In this case, only the changed input need be given for the following **start** directives.

5.4.14. **stop** termination

The keyword

stop

completes the input to an EDITAR run. It is not essential.

5.5. Sample Entries

The sample input given below is a collection of sample entries which is not intended to be a sample EDITAR run.

```
use 11,10   calc global   weight bilinear
print 1
buck b1 3.5e-3   search off
search on bsq for 0.98
fluxparm 150 0 1 1.E-5
output aus p1 11
output xs scan 7
groups 4 1 3 7 12 16
where some= 4(1)8
```

editar: 10

```
where cladfuel = fuel clad
where2d twopt  (1,10)  (10,1)
reqd cell some allmat1  u235  u235 @some
average u235 pu239
scanwt 7*0. 9*1.
smear 1 0
mixture fissile  u235 pu239 pu241
```

6. REFERENCES

[1]Robinson, G.S. [1986] - EDITAR: a module for reaction rate editing and cross-section averaging within the AUS neutronics code system. AAEC/E621.

[2]Rhoades W.A. & Childs, R.L. [1989] - DORT - Two-Dimensional Discrete Ordinates Transport Code. Oak Ridge National Laboratory, RSIC/CCC-484.

[3]Robinson,G.S. [1977] - AUS module MIRANDA - a data preparation code based on multiregion resonance theory. AAEC/E410.

[4]Bennett,N.W. & Pollard,J.P. [1967] - SCAN - a free input subroutine for the IBM360. AAEC/TM399.

CHAR — NUCLIDE BURNUP MODULE

1. INTRODUCTION

The AUS neutronics code system has been developed at Lucas Heights for application to a wide range of thermal and fast fission reactors. The three burnup modules CHAR, MICBURN and BURNMAC provide very flexible burnup capabilities for the AUS system. The basic burnup module is CHAR which solves the nuclide depletion equations for a number of reactor zones using an analytic technique. This method depends on ordering the nuclides such that, with few exceptions, burnup is in one direction within that hierarchy. This is not a real restriction on application of the method. The simplest use of CHAR is as one component of a lattice burnup calculation. However, the ability to handle multiple sets of cross-section data for a nuclide and to unsmear fluxes make CHAR suitable for global burnup calculations. CHAR may also be used for fission product inventory calculations which make use of a large set of fission products additional to the set for which cross sections are available on the standard cross-section library. An early version of the CHAR module, documented by Robinson [1], contains details of the method of solution of the nuclide depletion equations.

Global burnup involving CHAR is usually performed under the control of the MICBURN module which performs fuel movement and controls the calculation sequence. CHAR is then invoked by MICBURN to perform nuclide burnup and cross section mixing for each zone of the reactor. The few-group region-dependent cross sections for each nuclide which may be dependent on irradiation are obtained from previous lattice burnup calculations. Temperature dependence is not currently included.

MICBURN has largely replaced the BURNMAC module which is based on the assumption that the macroscopic cross sections are mainly a function of irradiation. Therefore global burnup calculations of reactivity and flux distributions can be performed using sets of macroscopic cross sections as a function of irradiation obtained from previous lattice burnup calculations. BURNMAC does not include any dependence of cross sections on temperature or flux level. In particular, it cannot follow reactivity transients induced by fission products.

2. DESCRIPTION

2.1. General

The CHAR module was developed to provide a method for solving the nuclide depletion equations which may be applied to any burnup calculation in which individual nuclides are represented. Any number of spatial divisions and any number of nuclides may be represented. The required data on neutron fluxes, geometry, cross sections, burnup processes, and initial nuclide concentrations are obtained from AUS data pools.

The module applies an analytic solution of the depletion equations to burn up each discrete material of the system under study; this system may be either a single lattice cell or a whole reactor. The user needs to ensure that there is in the data pools a separate definition in terms of its constituent nuclides for each material which, for burnup, may be considered to have a spatially uniform flux. The module adds the updated nuclide concentrations to the AUS data pool and optionally remixes the macroscopic cross sections.

2.2. Solution of the Nuclide Depletion Equations

At the heart of the calculation is the solution of the nuclide depletion equations for a given set of microscopic reaction rates using an analytic technique [1]. The solution requires that the reaction rates remain invariant (constant flux) and the nuclides be numbered such that nuclides burn to

form nuclides with a higher number. Nuclides may, however, burn to the immediately preceding nuclide so long as such occurrences are isolated. These restrictions cause no real problems in practice, and the constant flux assumption may be overcome by using a number of time steps and renormalising to a required power at the beginning of each step.

By using the standard nuclide ordering included in the burnup information on the data pools, only a few minor burnup reactions cannot be represented in the analytic solution. These are treated very simply by adding $\bar{N}_n\, a_{mn}\, \Delta t$ to the concentration of nuclide m at the end of a time step Δt, where a_{mn} is the microscopic reaction rate for producing nuclide m from nuclide n, $\bar{N}_n$ is the average concentration of nuclide n during the time step and $m < n - 1$. The analytic solution provides an analytic average concentration as well as the required concentrations at the end of the time step. The reactions requiring this simple treatment are noted in the printed output. With few exceptions, nuclides are ordered using an integer decimal of the form PZZAAAI,

where P is 1 for a fission product, else 0,
ZZ is the nuclide charge number,
AAA is the nuclide mass number, and
I is 1 for the ground state when a metastable isomer is also given, else 0.

The inclusion of alpha decay causes some problems. Thus ^{238}Pu is ordered after ^{242}Cm in order to properly represent ^{242}Cm decay to ^{238}Pu, and neutron capture in ^{238}Pu is represented only approximately.

The analytic method is susceptible to round-off errors. This causes few problems when 64-bit arithmetic is used but the inclusion of large sets of fission products requires some care. The problem arises when the fission products in the set are all interconnected. In calculations of fast reactors using the full set of fission products on the cross section library, it was advisable to break the interconnections by changing the burnup order indicator of ^{123}Te and ^{162}Dy to cause captures in these two nuclides to be treated approximately. Fast reactor calculations are more susceptible to this problem because one-group cross sections have less variation between nuclides than they have for thermal reactors. In calculations of fission product inventories, using the extended set of fission products described in Section 2.3, the approximate method of treating captures in ^{123}Te is built into the module.

2.3. Details Related To Data Pool Usage

The system geometry is obtained from an AUS geometry data pool on Fortran unit 4. The flux distribution for the geometry is obtained from any type of AUS flux data pool on Fortran units 25 or 26. This information is combined to give the volume and average group fluxes for each material explicitly represented in the geometry.

The STATUS data pools **st1** and **st2** should be given on Fortran units 9 and 10, respectively. The current burnup time is obtained from the **st2** file, and the information on the **st1** file for that time is read. Both the current time and the previous time must match unless the previous time on **st2** is zero. If a suitable **gfac** entry which gives the ratio of a k_{eff} to a k_∞ flux solution is present on **st2**, it may be used to modify the input flux.

The **st1** data give the current irradiation and composition of each discrete material and, if spatial smearing has been performed, the smearing factors and group condensation vector. A discrete material is simply one for which a definition in terms of nuclides is given on **st1**. The smearing factors give the ratio of fluxes and volume of the discrete material relative to the material into which it has been smeared. Any number of smearing/condensation stages may be performed when constructing the materials present in the geometry from the discrete materials. The group structure in which nuclide cross sections are given may be the same as that of the flux data pool or may correspond to a previous stage in the cross-section preparation.

Macroscopic cross sections are given on Fortran unit 7 in an AUS cross-section data pool. Microscopic data for the nuclides may follow the macroscopic data or be given on an additional unit, normally 8 . The cross-section data pool may contain several versions of a particular nuclide, since components of a material are identified by full 20-character names. The cross sections of a nuclide may also be dependent on irradiation. This dependence is included within one version of the nuclide. The burnup mechanisms are obtained from the data pool which includes a burnup table giving the decay constant, fission product yields, and the simple nuclide name of the product of decay and two possible production reactions.

The cross sections required from the file are absorption, fission and the production processes. The two production processes are normally (n,γ) and (n,2n) , although the production of two isomers by (n,γ) has also been represented. The absorption cross section in AUS is that which gives neutron balance. That is, the absorption is set to the neutron disappearance cross section (including fission) minus the (n,2n) cross section. To obtain a nuclide destruction cross section, the module normally adds to absorption twice the value of any (n,2n) cross section it can identify. When bilinear (flux and adjoint) weighting is used to form group cross sections, the destruction cross section should be set to the sum of the cross sections for fission and the two production processes.

In most cases the required burnup information is taken from the AUS data pool. However, it is possible to include further data on fission products in an additional file. This facility is intended for use in fission product inventory and decay heat calculations, whereas the standard data is intended for reactivity calculations. The additional file contains information on burnup, decay energy and gamma spectrum for a more complete set of fission products than that given in the cross-section library. When this file is given, the burnup information given there replaces that given in the AUS cross-section data pool. The cross sections for nuclides not given in the AUS data pool are taken to be zero. An additional file of 869 fission products has been generated from ENDF/B-VI and is available as the file *endf*6. *fisprodsp*. It also includes decay data for ten actinides.

After the burnup calculation, the **st2** file is restarted with an updated **time** entry which gives the current burnup time. A **time** entry, an **irad** entry giving irradiations of each discrete material, and an updated nuclide composition for each discrete material are added to the **st1** file. Any **$data** entries for the previous time which have the same module name as the previous **time** entry (normally the MIRANDA module) are added to **st1** with the material compositions, in the same order as before. If the option of remixing the macroscopic cross sections is specified, all entries in **st1** for the previous time are added to **st1** as it is assumed that the modules which wrote these entries are to be bypassed for the current time.

When cross-section remixing is specified, the updated macroscopic cross sections are written on Fortran unit 7. A material is remixed if all the nuclides comprising the material are present in the microscopic cross-section data pool and any required mixing rules are available. Otherwise the material is simply copied, and this fact is noted in the printed output. As for unsmearing, a material may have passed through a number of condensing and smearing stages.

3. INPUT SPECIFICATION

3.1. Data Layout

Input data is entered in free format on Fortran unit 1 using keywords to indicate the data type. Each keyword must begin on a new record and all data must be entered between columns 1 and 72. The module uses the input subroutine SCAN [2] to process the data. Default values have been set for most variables. A list of data following a keyword may be shortened if the defaults are satisfactory. The entries of each type are described in the following subsections. The

following conventions are used.

- Information to be reproduced exactly is given in **bold**.
- Items that the user will replace with an actual value are given in *italics*.
- Optional items are given inside [ ].
- Examples and defaults are given in `constant width` type.

Reference to an n-character word means up to n alphanumeric characters, of which the first is alphabetic.

3.2. Flux normalisation

The flux level at which the system is to be burnt may be set by specifying a volume integrated flux, flux, power, or power density for the whole or a part of the system. The format of the entry is

flux [density] *flux* **for** *name* **grps** *m* **to** *n*

or

power [density] *power* **for** *name*

where **density** is given only if a flux or power density is to be specified,

flux is the volume integrated flux (neutrons cm s^{-1}) or flux (neutrons $cm^{-2}s^{-1}$),

power is the fission power (W) or power density (W cm^{-3}),

name is **$all**, **$fuel** or a 20-character name (given as two 8-character and one 4-character words) of a discrete material in the system. **$all** means the normalisation is for the whole system; **$fuel** means the fuel materials; and a material name means the normalisation is for the named material only. **$all** is the default value,

m , *n* are the first and last energy groups of the nuclide group structure to be included in the flux normalisation. The default is all groups.

If a negative value is given for *power*, the absolute value is used as a multiplier of the input fluxes.

constant flux

is included with the **power** entry in order to burn for a number of time steps at a constant flux which corresponds to the given initial power. Otherwise the power renormalisation occurs at the start of each time step.

height *h*

gives the ratio of actual volume to volume represented in the flux calculation. For example, *h* might be the core height in cm in an XY calculation. The default value of *h* is 1.

Examples:

```
power  1.e+9
```

specifies a total power of 1000 MW.

```
flux  density  1.e+14  for  fuel, cella, orig  grps  5 to 5
```

specifies a flux level of 1.E+14 in group 5 for the named material.

3.3. Time Step

step *t n*

specifies *t* the time step interval in days, and *n* the number of steps of length *t* to be taken (default 1). As the solution of the depletion equations is analytic, the time step is that at which power renormalisation is carried out. That is, the time step is limited only by the requirement that the flux changes fractionally over the step. For burnup at constant flux, one step only is required unless intermediate printed output is desired.

The keyword **astep** provides an alternative method of specifying the time step for burning at "constant" power. When **astep** is used, the time step is divided into two intervals which result in the required energy release and also give the required power at the end of the step. This is done by determining the variation of power with time from the first interval, then adjusting the initial power and length of the second interval. The user may need to adjust the optional parameters if the specified time step is also to be obtained accurately. The input layout is

astep *step powrat tfract*

where *step* is the required time step (which actually sets the energy release),
powrat is the ratio of initial power to average power for the first time interval (default 1.0), and
tfract specifies the length of the first interval as a fraction of *step* (default 0.9).

3.4. Flux Data Pool

fluxlib fl*n m*

specifies which flux data pool, **fl1** or **fl2**, is used. The integer *n* has the value 1 for **fl1** on Fortran unit 25, or 2 for **fl2** on Fortran unit 26. The default value of *n* is 2. Any type of AUS flux data pool may be used. A flux data pool is not required for burnup at zero power. If *n* is zero, a one point burnup is performed and the cross-section data must be given for one energy group. The integer *m* is included for a multifile data pool from which no more than *m* files are to be read. The default is to use the last file. Thus if *m* is less than the number of files, the m^{th} file is used.

kinf

specifies that burnup is to be performed in a k_∞ flux. That is, any **gfac** entry on the **st2** file is ignored.

3.5. Cross Section and Burnup Data

xslib *n m*

specifies the Fortran unit number *n* of the microscopic nuclide cross-section data pool. The default is for microscopic data to follow the macroscopic data on Fortran unit 7. Where a separate data pool for nuclide data is given, this would normally be on unit 8. A separate data pool must be used if macroscopic data are to be remixed. A value *m* is specified in fission product inventory calculations to provide the Fortran unit number of a file on which additional fission product burnup data are given. The file *endf*6. *fisprodsp* provides such data.

addreac

is included if the destruction cross section is to be calculated as the sum of AUS cross sections 6, 7 and 8, *i.e.* fission and the two production reactions. This option is intended for use with bilinear weighted cross sections for which the standard option of calculating the destruction from the absorption cross section is not appropriate. With the current AUS *endfb*6 library, the use of **addreac** causes no loss of accuracy for any reactions in a thermal reactor.

remix

is included if the macroscopic cross sections are to be remixed after burnup is performed. Only those materials which undergo burnup are remixed. Even if the remixed cross sections are not required, this option may be needed in a complex cell calculation whenever the MIRANDA module is bypassed. For example, this may be required when consecutive links are made to the CHAR module to vary the time step taken.

remix 1

may be used to force remixing of all materials even if they do not undergo burnup.

ftype n

is included to change the fuel type for all nuclides to n. The fuel type on an AUS cross-section data pool specifies which of six possible sets of fission product yields is used. On current libraries, the six sets are for ^{233}U, ^{235}U, ^{238}U, ^{239}Pu, ^{240}Pu and ^{241}Pu.

inftype $(nuc_1\ n_1)\ (nuc_2\ n_2)\ \ldots$

is included to change the fuel type of specified nuclides. The data are given as pairs of nuclide name nuc_i and fuel type n_i.

fer f

is included to change the energy release per fission to f for all nuclides. The units of f may be either joules or MeV. The power in the CHAR module is fission power only. Hence the standard value for fission energy release includes a contribution from neutron capture.

infer $(nuc_1\ f_1)\ (nuc_2\ f_2)\ \ldots$

is included to change the fission energy release of specified nuclides. The data are given as pairs of nuclide name nuc_i and energy release f_i in joules or MeV.

noactd

is included to exclude the actinide decay data on files such as *endfb6*. *fisprodsp* from the calculation. This enables a true fission product decay term to be calculated if this is required, rather than the normal option of including the decay of some actinides in the results.

noupdat

is included to prevent any updating of AUS data pools. Neither the status nor the cross section data pools are updated. This enables the results of a burnup perturbation to be printed without affecting the main calculation. For example, this may be used to print the results of a period of cooling, *i.e.* operation at zero power, at each step of a burnup calculation at power.

3.6. Irradiation Dependent Nuclide Cross Sections

The microscopic nuclide data, which is normally on Fortran unit 8, may be a type-4 AUS cross-section data pool in which irradiation dependence replaces dependence on effective scattering cross section on a type-2 data pool. The irradiation values at which the data are tabulated are not entered in the data pool but are given by the following entry.

tabled *cell* $n\ r_1, r_2, \cdots, r_n$

where *cell* is the 4-character cell-name for which data are tabulated as a function of irradiation,

n is the number of irradiation points,

r_i are irradiation values in W day cm^{-3}.

It should be noted that a cell-name forms part of a full 20-character material or nuclide name, and that nuclide cross sections for all materials of a lattice cell are tabulated against irradiation. The irradiation values are those for the (first) fuel material in the lattice cell, which are the values printed out by CHAR in a previous burnup calculation in which the nuclide data were prepared. For a given material, the irradiation value used to interpolate in this table is the first non-zero irradiation value of any material for which characters 9 to 20 of the material name are the same as those of the given material. Thus the fuel, can and coolant data in a cell may all depend on the fuel irradiation value.

3.7. Dimension Data

The only dimension data possibly required is the maximum number of burnable materials. Other data used for dimensions are obtained from the various data pools.

maxm m

is included to alter the maximum number of burnable materials from the default value of 500.

3.8. Printed Output Options

print n m

is included to control that section of the printed output not edited by nuclide (Section 3.9). If $n \geq k$, printed output of type k is produced where the print types are

0 a minimal print of materials and irradiations for each time step,
1 a print of concentrations and time average concentrations at the end of each time step for each nuclide for each material,
2 a print of group fluxes for each material, and
4 a print of one-group microscopic reaction rates for each nuclide for each material.

The default value of n is 1 . The integer m is intended for use in fission product inventory calculations using additional fission product data. The values for m are

0 nuclide concentrations are printed in units of 10^{24}atoms cm^{-3},
1 nuclide concentrations are printed in grams,
2 rather than the print of type `n=1` above, a full nuclide print including nuclide activities and decay power is produced. The total photon spectrum due to fission products and actinides is also printed.

The default value of m is 0.

3.9. Editing Facilities

Editing facilities are available to provide printouts of the concentrations of selected nuclides and the reactions for mixtures of nuclides. Spatial averages of the above quantities may also be printed. The print of reactions gives loss (absorption plus decay), fission and capture for the selected mixtures in units of events per $10^{-24}cm^3$ over the burnup step. The edit facilities would normally be used in conjunction with a `print 0` entry.

follow n nuc_1 nuc_2 ... nuc_n

provides n the number of nuclides for which concentrations are to be printed, and nuc_i the set of named nuclides.

reactions n $(m_1$, nuc_{11} nuc_{21} ... $nuc_{m_1 1})$ ... $(m_n$, nuc_{1n} nuc_{2n} ... $nuc_{m_n n})$

provides n the number of mixtures to be formed, m_i the number of nuclides in the i^{th} mixture, and nuc_{ji} the nuclide names in the i^{th} mixture. The nuc_{ji} must be taken from the set named in **follow**.

average n $nama_1$ $namb_1$ $nama_2$ $namb_2$ ... $nama_n$ $namb_n$

provides n the number of spatial averages to be formed, and the two 4-character names $nama_i$ $namb_i$ which give the first halves of the first two 8-character words of the full material names to be included in the i^{th} average. The names may also have the values **$all** which means everything or **$fuel** which means all fuel materials. The materials which have the specified half-names are included in the average. The use of **average** requires an appropriate convention for forming material names. Average results are printed for both concentrations and reactions.

char: 8

Example:

```
follow  6  u235  u238  pu239  pu240  pu241  pu242
reactions 4 3 u235 pu239 pu241 2 u238 pu240 1 pu239 1 pu240
average  3  fuel core  fuel blnk  fuel $all
```

would form averages for the core, blanket and whole reactor from the materials
`(fuel,core01), (fuel,core02), (fuel,blnk01), (fuel,blnk02)`
for two core cells labelled `core01` and `core02` and two blanket cells labelled `blnk01` and `blnk02`.

3.10. Irradiation and Fluence Monitoring

The irradiation density (W days cm^{-3}) of each discrete burnable material is updated by CHAR and included in the **irad** entry on the **st1** file. A number of other irradiation or fluence monitors may also be included in the **irad** entry for use by modules such as MICBURN. The required CHAR input is

fluence nuc_1 nuc_2 ...

where nuc_i is the 8-character name of a nuclide which is present in appropriate discrete materials.

For example, if an insignificant amount (say 1.e-20) of `zz999` is included in each material and
`fluence zz999`
is included in CHAR, then values of fluence are calculated as the time integral of nv_0, the product of neutron density and a velocity of 2200 m s^{-1}.

3.11. START

start

is included to cause the calculation set up by the previous entries to begin.

4. REFERENCES

[1]Robinson,G.S. [1975] - AUS burnup module CHAR and the associated STATUS data pool. AAEC/E372.

[2]Bennett,N.W. & Pollard,J.P. [1967] - SCAN - a free input subroutine for the IBM360. AAEC/TM399.

MICBURN – MODULE TO CONTROL REACTOR BURNUP USING MICROSCOPIC NEUTRON CROSS SECTIONS

1. INTRODUCTION

Before the introduction of MICBURN, the modules used for burnup within the AUS neutronics code system were CHAR and BURNMAC. The CHAR module may be used to perform burnup of a cell representing a single fuel element or a complete reactor, using an arbitrary set of nuclides for which microscopic cross sections and burnup mechanisms are available on an AUS cross section file. The descriptions of materials are taken from an AUS status file while the geometry and flux are taken from AUS geometry and flux files. The connection between the material names on the status file and the geometry file is provided by the AUS macroscopic cross section file used in the flux calculation. The action of CHAR is to update the nuclide concentrations on the status file and optionally remix the macroscopic cross sections. CHAR has been used mainly for cell calculations because a module to manipulate the status file and thus control a reactor calculation has not been available. The MICBURN module supplies that function.

Most reactor burnup calculations in AUS have been performed using the BURNMAC module which is based on the assumption that the macroscopic neutron cross sections of reactor core components are mainly a function of irradiation. Therefore global burnup calculations of reactivity and neutron flux distributions can be performed using sets of macroscopic cross sections as a function of irradiation obtained from previous cell burnup calculations. BURNMAC is a simple module to interpolate cross sections, make changes in the reactor loading and perform fuel accounting. MICBURN provides an advance on BURNMAC by assuming only that the region-dependent few-group microscopic cross sections of nuclides in a fuel element are a function of irradiation. A preliminary cell burnup calculation is still required to provide these microscopic cross sections and other details of the cell. However, nuclide reaction rates which determine burnup are calculated using the space-dependent neutron spectrum in the reactor rather than the cell spectrum as in the BURNMAC method. The main advantages over BURNMAC are the improved accuracy in neutron fluxes and reactivity resulting from the less stringent assumption, the ability to calculate nuclide loadings in the fuel directly and the ability to make calculations dependent on the neutron flux level (*i.e.* treat fission product transients).

The function of MICBURN is similar to HIFUME [1] but HIFUME is restricted to a particular two-group XY model of ANSTO's research reactor, HIFAR, whereas MICBURN is quite general. It may be applied to one, two or three dimensional reactor models in any number of neutron groups with any type of fuel element. It does have a few options which have been included to meet the particular requirements of HIFAR modeling. Many of the MICBURN input directives have been made similar to BURNMAC and the two modules should be regarded as having complementary functions.

2. GENERAL DESCRIPTION

The first run of MICBURN for any particular calculation serves to initialise the reactor model to be used. Firstly, the representation of each type of fuel element is obtained from the status file written in the preliminary AUS cell calculation which generated microscopic cross sections for all nuclides in the cell as a function of irradiation. The user must ensure that these microscopic cross sections include the cell smearing factors (use the EDITAR module option `smear 0 -1`) and are in the form produced by the JOINER module. The nuclide cross sections for all cell types must be combined on one AUS cross section file which is read from Fortran unit 11 by MICBURN. The irradiation values at each cell burnup step and the rules for generating cell cross

sections from the nuclide cross sections are obtained from the status file generated in the cell calculation. In order for the rules to be available, the user must ensure that the cell burnup calculation generates macroscopic cross sections for the cell, even if these are not required otherwise. There may be a number of fuel materials and other burnable materials within a fuel element and this detail is preserved throughout the MICBURN calculation. However, users should be careful not to complicate the calculation by including multiple fuel materials when the difference between them is not significant. Since the burnup of each distinct fuel material is calculated at each zone of the reactor, the number of burnup calculations may become excessive. The required cell status files are named in the MICBURN input.

The second function of initial runs is to write the description of the reactor entered by the user as an AUS geometry file on Fortran unit 4. This description includes the layout of fuel element positions in the reactor core and the division of the positions into burnup zones in which the flux is assumed to be constant in determining burnup. For calculations in which fuel elements are not explicitly represented, the user should consider each required burnup zone to be a 'fuel element'. Normally, this geometry data file would be used by the flux calculation module. Where the reflector geometry is complicated, the flux module might not use this data file, but it is still required by MICBURN.

The last function of a first run is to initialise the nuclide compositions of each fuel element in the core. This may be done by simply loading new fuel through the core or by using the cell status files to provide nuclide compositions at requested burnup values. Dummy macroscopic cross section and flux files are written on Fortran units 10 and 25 respectively, so that the first call to the CHAR module can be used to mix the required macroscopic cross sections. The cross sections are written in the order of cross sections for each zone for each fuel element position in turn followed by reflector cross sections.

The main functions of MICBURN are to control the sequence of the calculation, to change fuel, and to provide summary printed output as the calculation proceeds. Control of the overall AUS calculation is exercised by MICBURN rather than by an AUS path program. To achieve this, a particular form of path program is required. The following path, which may be obtained as `step micburn` from the AUS path library, shows the requirements.

```
      alter (12,f11)
10    link micburn (1,dd2), (4,gm1), (8,st2), (9,st1), (10,xs1), (11,xs2),
 1    (12,dd9), (13,dd42), (14,dd3), (15,dd4), (16,dd5), (25,f11), (6,dd6)
      kcond = icond
      if( kcond .ge. 4 ) link char (1,dd4)
      if( kcond .ge. 2 ) then
      write(12) 0
      rewind 12
      endfile 12
      link pow3d (1,dd3)
      endif
      if( mod(kcond,2) .eq. 1 ) link char (1,dd5)
      go to 10
      end
```

The linking of modules is controlled by the condition code (icond) set by MICBURN. The input to the CHAR module which is linked for mixing only (**dd4**) or for burnup (**dd5**) is generated by MICBURN and written on Fortran units 15 and 16 respectively. The flux calculation is normally performed using the POW3D module as indicated. MICBURN may be used to modify the input to the flux module as the calculation proceeds, if that input (**dd3**) is available on Fortran unit 14.

Information on the current state of the calculation, particularly the nuclide composition of each part of the reactor core, is passed between modules on the pair of AUS status files **st1** and **st2** which are on Fortran units 9 and 8 respectively in MICBURN. As well as the standard status entries processed by CHAR, MICBURN adds its own special entries to pass information from one MICBURN link to the next.

Once the model used in a particular calculation has been checked, the user should not need to refer to any output except that produced by MICBURN. Hence, the standard output on Fortran unit 6 in MICBURN is directed to the file **dd6** in the MICBURN path program and the output from the rest of the AUS calculation may be treated as an error file.

MICBURN includes a comprehensive set of fuel changing directives. As well as simple loading of new fuel elements, there are directives to rearrange the current core loading and to reload elements previously discharged. To enable this reload feature, the contents of each element discharged from the core is written as a Fortran formatted file on unit 12 (**dd9**). This file should be useful as part of any fuel accounting system which keeps track of spent fuel elements.

The standard (printed) output from MICBURN always includes a description of actions taken, loadings of important nuclides in fuel elements as they are loaded or discharged, and reactivities calculated by the flux module. The optional output, which may be produced at any time, includes a map of power and irradiation in each zone, loadings of important nuclides in each fuel element, full details of the loading of selected fuel elements and a traditional HIFAR style printout.

As MICBURN is intended to be used for long-term calculations following many refuelling cycles, methods for continuing the calculation and recovering from errors are included. These are discussed in detail later but the error recovery procedure involves the use of the file **dd42** on Fortran unit 13 as a partial copy of the **st1** file. It is assumed in all restarts that all the files **gm1**, **st1**, **xs1**, **xs2**, **dd9**, **dd42**, **fl1** have been kept from the previous run. For efficiency, the **fl3** file should also be kept.

3. INPUT SPECIFICATION

3.1. Data Layout

The input data are read using the SCAN input routine [2]. The data, entered in columns 1 to 72, are in the form of a number of entries each of which consists of a keyword followed by a string of data items. In most cases, more than one entry may be given on each input line. The entries are of six types: fuel and reflector specification, geometry description, option specification, program identification, fuel movement directives, and control directives. The entries of each type are described in the following subsections. The following conventions are used.

- Information to be reproduced exactly is given in **bold**.
- Items that the user will replace with an actual value are given in *italics*.
- Optional items are given inside [ ].
- Examples and defaults are given in `constant width` type.

3.2. Fuel and Reflector Specification

These entries are normally given first in the initial link to MICBURN. However to allow for the possibility of new fuel types being introduced, additional fuel entries may be given first in subsequent links. A **reflector** entry is required if the reactor calculation contains a reflector. There are two possible options for the treatment of reflector materials. Firstly, they may be of no concern to MICBURN because the cross sections are obtained by POW3D from a separate file. In this case, only the number of reflector materials used in the geometry description is required. Secondly, the cross section data may be given as microscopic data on the same file as that used for

the fuel. The data are mixed by CHAR according to the definitions included in the MICBURN input. This method should be used if the reflector data is to change, *e.g.* in order to represent heavy water degradation. The reflector entry is:

reflector *nrmat* [*refnam*]

where *nrmat* is the number of reflector materials,
refnam is an alphanumeric label of up to 8 characters (A8) used on the **head** entry in MIRANDA when generating the reflector cross sections.

The reflector entry may be followed by a number of **defn** entries to define the reflector materials. These entries have exactly the same form and meaning as in MIRANDA. That is:

defn *matnam* $nucl_1$ $conc_1$ $nucl_2$ $conc_2$ ...

where *matnam* is the name (A8) of a reflector material,
$nucl_i$ is the name (A8) of a reflector nuclide on the cross section file,
$conc_i$ is the density of $nucl_i$ in 10^{24} atom cm^{-3}.

A **fuel** entry is given for each type of fuel in the reactor. It has the form:

fuel *ftnam statfile*

where *ftnam* is an alphanumeric 4-character label,
statfile is the name of the AUS status file generated in the cell burnup calculation which prepared the microscopic cross sections for fuel of this type. It may be a simple file name if the file is in the current working directory or a full path name for the file.

The label *ftnam* is used in identifying the type of each fuel element loaded. Each **fuel** entry may be followed by a number of **defn** entries which modify the default initial loading of the specified nuclides for a fuel element of that type. If no **defn** entries follow, the default initial loading is that used in the cell calculation. The option to modify the default initial loading would not be used normally, but could be used if the same set of microscopic cross sections were suitable for two fuel types with slightly different initial loading.

3.3. Geometry

Geometry data are given only once in an initial MICBURN run and must follow immediately after the fuel and reflector data. The data specify the intervals and boundary conditions used in the flux calculation, the layout and labeling of fuel element positions, the division of fuel elements into burnup zones, and the layout of reflector materials. Where appropriate, the data entries have the same form as those used in POW3D.

The mesh intervals and boundary conditions are given exactly as in POW3D using **xm, ym, zm, rm, rm(sphere)**, as appropriate, where each entry is of the form

xm d_L , δ_1 , δ_2 , ... , δ_n , d_R

where d_L, d_R are left and right boundary conditions and δ_i is the i^{th} mesh interval of a set of n intervals.

As in POW3D, **reg** entries are used to specify the reflector material layout and may also be used to specify the layout of fuel element positions. The form is

reg **mx** i_1 , i_2 , ... , i_I **my** j_1 , j_2 , ... , j_J **m**(m)

or

reg **mr** i_1 , i_2 , ... , i_I **mz** j_1 , j_2 , ... , j_J **m**(m)

or

reg **mx** i_1 , i_2 , ... , i_I **my** j_1 , j_2 , ... , j_J **mz** k_1 , k_2 , ... , k_K **m**(m)

where the i, j and k are interval numbers, m specifies the m^{th} reflector material for $m > 0$, and $|m|$ specifies the $|m|^{th}$ fuel element position for $m < 0$. Use of **reg** entries becomes tedious for large numbers of fuel element positions. An alternative specification in terms of channels may be used for regular layouts in which the element positions are bounded by a number of planes. The entries are

lx mx o_1 , o_2 , . . . , o_{mx+1}

ly my p_1 , p_2 , . . . , p_{my+1}

lz mz q_1 , q_2 , . . . , q_{mz+1}

where mx is the number of channels in the X or R direction, the intervals k satisfying $o_i \le k < o_{i+1}$ are in the i^{th} channel in the X or R direction, LY is used for the Y direction (or the Z direction in RZ geometry), and LZ is for the Z direction in XYZ geometry. The fuel element positions are then numbered (on a printed page) from left to right and top to bottom for each XY channel which does not contain reflector material. In 3D geometry, each fuel element would normally extend over the full core height.
Example:

```
reg    mx  1(1)8    my  1(1)8    m(1)
reg    mx  1(1)6    my  1(1)6    m(0)       ... to clear
reg    mx  5 6      my  5 6      m(1)
lx 3   1 3 5 7    ly 3   1 3 5 7
```

The result is a definition of eight fuel element positions, each with a 2×2 mesh:

```
.  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .
1  1  2  2  .  .  .  .
1  1  2  2  .  .  .  .
3  3  4  4  5  5  .  .
3  3  4  4  5  5  .  .
6  6  7  7  8  8  .  .
6  6  7  7  8  8  .  .
```

The fuel element positions may be given labels by which they are referenced in fuel movement directives. This is done using

positions pos_1 , pos_2 , . . . , pos_n

where pos_i is a 4-character alphanumeric label for the i^{th} fuel element position, and
n must equal the number of fuel element positions.

The first character of each pos_i must be alphabetic. The default labels are `p1` to `pn`.

In calculations without explicit representation of fuel elements the user should consider burnup zones to be 'elements'. Where fuel elements are explicit, the default option is one burnup zone per element. Division of an element into burnup zones is obtained using

zonereg nx ny nz

where nx is the number of spatial mesh intervals per element in the X direction,
ny similarly specifies the second (Y or Z) direction, and
nz similarly specifies the third (Z) direction.

The **zonereg** entry is followed by a second set of **reg** entries which specify the division of a fuel element into burnup zones. In these **reg** entries, the number of interval numbers following **mx** is

nx, the other directions are similar, and *m* specifies the zone number for the set of mesh intervals instead of a material for standard POW3D **reg** entries. The maximum value of *m* determines the number of burnup zones per element. The division of an element into zones is usually required only in 3D calculations and printed output is produced on the expectation that the zones are axial zones numbered from bottom to top.

In 1D and 2D calculations, the missing dimensions should be represented using the directive:

height *h*

where *h* is the ratio of actual volume to volume represented in the flux calculation.
Thus, in an XY model, *h* would be the core height (cm).
In an XYZ or RZ model reflected at the core midplane, h would be 2.

The above entries completely describe the geometry. The initial loading of fuel elements is performed using the fuel movement directive **refuel** described below.

The CHAR module allows a number of nuclides to be used in forming reaction rates which are to be used as detectors in calculating the neutron fluence. MICBURN makes use of this feature to allow the program-average neutron flux to be calculated throughout the core. The nuclides which are to be used as flux detectors are specified using:

detector $denam_1$ $denam_2$ ...

where $denam_i$ is the name (A8) of a nuclide on the cross section library.

For this option to function correctly, there needs to be a nuclide on the microscopic cross section file with fully qualified name of ($denam_i$, *cellnam*, **orig**), in layout (A8, A8, A4), for each fuel type. Here, *cellnam* is an alphanumeric label used on the **head** entry in MIRANDA when generating the cross sections for that fuel type. To obtain thermal (nv_0) fluxes in a component of the fuel elements, cross sections for the AUS pseudo nuclide `zz999`, which has cross sections inversely proportional to neutron velocity, should be generated for that component of the fuel element cell. As for normal nuclide cross sections, the EDITAR option `smear 0 -1` should be used for detector cross sections.

3.4. Option Specification

This rather miscellaneous collection consists of those directives which would normally be specified only once or infrequently. However, they may be specified at any stage of the calculation.

To modify the composition of reflector materials or materials in the core which do not undergo burnup, the directive used is:

afrac *matnam* $nucl_1$ $afrac_1$ $nucl_2$ $afrac_2$ ...

where *matnam* is the name (A8) of such a material,
$nucl_i$ is the name (A8) of a nuclide originally in that material,
$afrac_i$ is the new atom fraction of $nucl_i$ in the material.

This directive was included to allow for degradation of heavy water. *e.g.*

```
afrac   d2om   d2o   0.995   h2o   0.005
```

would change the composition of the material `d2om` anywhere in the reactor to 0.5% light water.

To modify the composition of the fuel materials for all fuel elements in the core by removing some nuclides, the directive used is:

nonuc $nucl_1$ $nucl_2$ ...

where $nucl_i$ is the name (A8) of a nuclide present in the fuel material. The concentration of the listed nuclides is set to zero.

This directive was included to allow perturbation calculations which determine the reactivity worth of one or more nuclides. *e.g.*

```
nonuc    xe131    xe133    xe135    kr83    kr85
```

would remove all noble gases from the fuel so that their worth could be calculated.

In research reactors, it is usual to allow for slight variations in the initial fuel loading of each fuel element. To enable this to be done readily, the user may define a mixture of nuclides for which the total loading is to be specified for each fuel element. The directive is:

wtfrac *mixnam* $nucl_1$ $wfrac_1$ $nucl_2$ $wfrac_2$ ...

where *mixnam* is the name (A8) of a nuclide mixture,
$nucl_i$ is the name (A8) of a fuel nuclide on the cross section file,
$wfrac_i$ is the weight fraction of $nucl_i$ in the mixture.

The quantities which may be given in specifying the initial loading of a fuel element are established using the directive:

fspec *ftnam* $namm_1$ $namm_2$ ...

where *ftnam* is the name of the fuel type to which the specification applies,
$namm_i$ is the name (A8) of a mixture defined using **wtfrac** or a fuel nuclide.

The **fspec** directive merely sets the list of quantities which may be specified when individual elements are loaded. Thus, in order to be able to specify the loading of ^{235}U and remaining uranium in fuel elements of fuel type u423 , the entries might be:

```
wtfrac  ufer  u234  .023  u236  0.26  u238  0.717
fspec  u423  u235  ufer
```

MICBURN produces summary information on fuel elements as they are loaded and discharged, and for the whole core on request. The user may specify those nuclides or elements in which they are interested by using the directive:

track $trknam_1$ $trknam_2$ ...

where $trknam_i$ is the name (A8) of a fuel nuclide or element which is to be included in these summaries. The default values are:

```
track    u235    u    pu239    pu
```

As well as the value of k_{eff}, the excess reactivity (%) of the system is reported following each flux calculation. On some research reactors, the excess reactivity is often reported as the equivalent excess reactivity in a core with a standard fissile mass. The directive required to give such a scaled reactivity is:

rhomass *smass* *exp* *rhof* $nuct_1$ $nuct_2$ ...

where *smass* is the standard fissile mass (kg),
exp is the exponent used in scaling the reactivity,
rhof is a divisor used in scaling the reactivity,
$nuct_i$ is a nuclide to be included in calculating the fissile mass which must be one of the nuclides on the **track** entry.

Thus for a standard mass of 2.75 kg, the entry to use the total mass of ^{235}U and ^{239}Pu might be:

```
rhomass 2.75 0.75 1. u235 pu239
```

The expression used in calculating the scaled reactivity ρ_S from the reactivity ρ for fissile mass M is:

$$\rho_S = \frac{\rho \times (M / smass)^{exp}}{rhof} \quad .$$

One set of values reported when the HIFAR print option is used, is an approximate value for the increase in reactivity resulting from the replacement of a current fuel element by a new element for each position in the core. These values are based simply on input values giving the loss in reactivity per gram of ^{235}U burnt for each position in the core. To enter these values, the directive required is:

rhofuel $rhob_1$ $rhob_2$ ... $rhob_n$

where $rhob_i$ is the reactivity (%) loss per gram in the i^{th} element,
n is the number of fuel element positions.

An option has been included to allow the calculated neutron flux to be modified by a factor dependent on neutron group. One possible use for this option arises in XY calculations using a core centre-plane model. This option may then be used to provide the ratio of axial average to centre-plane fluxes for use in calculating reaction rates. The directive is:

gfac $facg_1$ $facg_2$ ... $facg_{ng}$

where $facg_i$ is the factor applied to the i^{th} neutron group,
ng is the number of neutron groups.

When adjoint weighting is used to prepare the microscopic cross sections, this fact must be made known to CHAR so that an appropriate destruction cross section will be used. The MICBURN directive required is:

adjoint

MICBURN uses the automatic time step adjustment option (**astep**) of CHAR which divides a time step into two intervals to give the required time integrated power and power level at the end of that time step. There are two parameters which give the ratio of initial power to specified power, *powrat* , and fractional length of the first interval, *tfract* . If a variation from the default values of 1.0 and 0.9 is required, the required MICBURN directive is:

astep *powrat* *tfract*

3.5. Program Identification

To identify the current reactor operating program or cycle, the directive used is:

program *nprog*

or

cycle *nprog*

where *nprog* is an integer which identifies the reactor operating program of following directives.

This directive provides a means of labeling an operating program and gives some structure to the entire calculation. Its relevance in restarts will be considered later. The burnup time and all program average quantities are reset to zero. Though any mixture of directives may be used within a program, the following structure should be used to make best use of the printed output. Each program is considered to consist of a number of fuel changes, which may be interspersed with burnup steps at zero power and calculations of k_{eff}, followed by a number of burnup steps at any power and k_{eff} calculations but no further fuel changes. If fuel changes are made in a program after burnup at power, some program-average printed quantities (*e.g.* power) will be incorrect.

3.6. Fuel Movement Directives

The module provides facilities for loading fuel elements, changing their position in the core, and also saving elements for later reloading. A printed summary of the contents of elements unloaded and loaded is given at the completion of a set of fuel movement directives. The loading directive is

refuel *n* [**delete**]

where *n* is the number of fuel elements to be loaded,
delete is given if unloaded fuel elements need not be saved for reloading or for the user's accounting system to access.

This directive is followed by *n* **load** or **reload** directives, each of which specify a fuel element that is to be loaded in a nominated position. The directives are:

load *pos type* [*num* [fsg_1 fsg_2 ...]]

reload *pos type num*

where *pos* is the element position label,
type is the fuel type which must be identical with one of the values *ftnam* on a **fuel** entry,
num is a 4-character alphanumeric fuel element number,
fsg_i is the initial loading in grams of the i^{th} nuclide or mixture on the **fspec** entry.

If any fsg_i values are given, the number of values must match that on the **fspec** directive for *ftnam*. The default initial loading is that taken from the cell burnup calculation, perhaps modified using **defn** entries. If no element number is given (and hence no initial loading can be given), the default is one more than the last number generated by the module. For use with the initial fuel loading in particular, a **load** directive may be followed immediately by a **burnt** directive to specify the burnup of a partially burnt element to be loaded. The form is:

burnt [**grams**] rad_1 rad_2 ...

where **grams** is given if the following values specify ^{235}U content in grams, else they specify Watt-days,
rad_i are 'irradiation' values. If only one value is given, it is used for all axial zones. Otherwise, the values are for each axial zone. The units for irradiation values are 'per element' whether one or more values are given.

The position of fuel elements in the core may be completely respecified using

rearrange pos_1 pos_2 ... pos_n

or

rearrange no typ_1 num_1 , typ_2 num_2 ... typ_n num_n

where pos_i is the position label currently occupied by the element to be moved to the i^{th} element position,
n is the number of elements in the core,
typ_i and num_i specify the type and number of an element in the core to be moved to the i^{th} element position.

Alternatively, movements from one element position to another may be specified using

move (i_1 , j_1) , (i_2 , j_2) , ... , (i_m , j_m)

which causes the elements in positions i_1 to i_m to be moved to positions j_1 to j_m respectively. A **move** directive should not result in positions not containing an element.

3.7. Control Directives

This set of directives controls the flux calculations and burnup steps to be undertaken by the module. The power level to be used in following burnup calculations is specified using:

power p

where p is the power level in watts. Quite different action for burnup directives is taken by the module depending on whether the power is zero or non-zero.

A flux calculation of the current reactor state may be initiated using either of the directives:

flux

or

keff

Either of these directives should be included whenever the k_{eff} (or reactivity) is required, unless a burnup step at power follows. The flux will automatically be calculated under these circumstances if it is necessary. The directives which initiate burnup are:

step Δt

or

burn Δt

where Δt is the time step in days. These cause burnup to be performed immediately at the previous specified power level.

Since it may be necessary to change the input to the flux module as the calculation proceeds, the following facility is provided.

replace nl

...

where ... represents nl lines of input which are used to replace existing lines in the input data for the flux module, and columns 73 to 80 must be identical in an existing line and the replacement line.

The three following directives also modify the input to the flux model but they are specific to HIFAR. They may be used to modify the POW3D input data in the standard 2D and 3D models of HIFAR with 30×30 mesh intervals in the XY plane [1]. They depend on a particular layout of data in that model and standard HIFAR facility names.

corerigs ncr lab_1 c_1 , lab_2 c_2 , ... , lab_{ncr} c_{ncr}

reflrigs nrr lab_1 c_1 , lab_2 c_2 , ... , lab_{nrr} c_{nrr}

where ncr is the number of rigs in the reactor core,
nrr is the number of rigs in the reflector,
lab_i is the 4-character label of the facility in which the i^{th} rig is to be included,
c_i is the concentration of thermal poison used to represent the i^{th} rig.

The two directives function differently in that **corerigs** gives the complete set of rigs to be included in the core whereas **reflrigs** gives changes to be made to the current state of the reflector. Additionally, the Coarse Control Arm (CCA) angle may be set in the 3D model using:

cca *angle*

where *angle* is the CCA angle in degrees.

Printed output to be produced is initiated using the directive:

print [**map**] [**loading**] [**hifar**] [**fe** [*typ* *num*] [**all**]]

where **map** produces a map of the program-average power and current irradiation in each fuel element and (in 3D cases) each axial zone,
loading produces a summary of the nuclide loading in each fuel element in the core,
hifar produces a traditional HIFAR summary of the current program,
fe produces details of all nuclides in a specified fuel element,
typ and *num* are the fuel element type and number,
all may be given rather than *typ* and *num* if details on all fuel elements in the core are required.

A selection of the required options may be made. A HIFAR type print should be requested only at the completion of an operating program as it is meant to summarise the program. It includes a number of items for each fuel element including power and neutron flux levels. The nuclides in the **loading** type print are those specified using the **track** directive. The **fe** type print was included for code testing but may be useful in other circumstances.

The directive

stop

is used to terminate the current calculation. An end-of-file serves the same purpose. Restarting from a previous run is considered in the next subsection.

3.8. Restarts and Error Recovery

It is intended that MICBURN should be easy to use in long-term reactor calculations following many refuelling cycles. Therefore a number of restart options are provided. They all rely on the files **gm1**, **st1**, **xs1**, **xs2**, **dd9**, **dd42**, **fl1** being kept from one run to the next.

The first option is to carry on from the end of the previous run. This is the default which requires no special input or action. The facilities of the AUS supervisor AUSYS may be used to restart from a nominated time within the current operating program, but this procedure is prone to user error and is not recommended.

The remaining options rely on the directive

save

which should be included at the end of an operating program once the details of that cycle have been finalised. When the first input directive is **program**, MICBURN will automatically restart from the end of the previous program if a **save** directive was included at the end of that program. The normal procedure should be to investigate the requirements of the current program (and possibly the next) in a number of runs which each restart from the last **saved** program. Then the **save** directive can be included for that program and attention shifted to the next. To enable additional **fuel** entries to be included when using this procedure, a dummy **program** directive with minus the required program number should be given first. This is then followed by the **fuel** entries and the actual **program** directive.

To enable recovery from an error which was introduced in an earlier program for which **save** has been issued, the first input directive may be

recover *nprog*

where *nprog* is the number of the last correct program.

Again, this option requires that a **save** directive was included at the end of that program. The following input data should then begin from the **program** directive of the next program *i.e.* the first to be corrected.

These restart options are made possible by a number of manipulations of the files **dd9**, **st1** and **dd42** which are used for the discharged fuel element file, the status file and the save-status file respectively. Any elements which have been discharged are added to the discharge file and

deleted from computer memory when a **save** directive is encountered. Any reloaded elements are not deleted from the file, which may therefore contain more than one entry for such an element. The main file used to store information is the status file in which a history of the current program is maintained. This is achieved by writing the current status of the calculation on the end of the status file on each exit from MICBURN and CHAR. The **save** directive results in a normal write to the file but the file is restarted before the data is written. Thus knowledge of previous times is lost on this file. The information written on the status file by the **save** directive is also added to the save-status file. The actions of **recover** are to restart the status file with the data on the save-status file for the requested program and delete data for the succeeding programs from both the save-status and discharged-fuel files. This should provide a complete error recovery procedure. The only additional actions which the user may need to perform are periodically to archive the save-status file when it becomes too large and then to restart the file from scratch.

Provided that the **save** directive is included, a fuel accounting program used in conjunction with MICBURN can obtain details of discharged elements from the discharged fuel file. If fuel elements are reloaded, the program accessing the file needs to make allowance for the occurrence of multiple entries for one element. When **save** is included, the details of elements in core are written on Fortran unit 17 in the same format as the discharged fuel file. This is intended to assist in fuel accounting.

4. REFERENCES

[1]Storr, G.J. [1989] - HIFUME - a fuel management code for HIFAR. ANSTO/NTP/TN132.

[2]Bennett, N.W., & Pollard, J.P. [1967] - SCAN - A free input subroutine for the IBM360. AAEC/TM399.

BURNMAC – MODULE TO PERFORM REACTOR BURNUP USING MACROSCOPIC NEUTRON CROSS SECTIONS

1. INTRODUCTION

BURNMAC is a burnup module of the AUS neutronics code system which is based on the assumption that the macroscopic neutron cross sections of reactor core components are mainly a function of irradiation. Therefore global burnup calculations of reactivity and neutron flux distributions can be performed using sets of macroscopic cross sections as a function of irradiation obtained from previous lattice burnup calculations. BURNMAC is a simple module to interpolate cross sections, make changes in the reactor loading and perform fuel accounting. It is used in combination with a flux calculation module to undertake this type of burnup calculation. Since the dependence of cross sections on temperature or flux level is not included, it is not well suited to calculations of large thermal power reactors where such effects may have a considerable effect on flux distributions. BURNMAC is a revised version of original code [1] which has been retained with the name BURNMAC1 because the input to the two versions is incompatible. BURNMAC has been extended to three spatial dimensions; the input has been revised to allow for calculations extending over many reactor cycles with many changed fuel elements; and some features specific to modeling the HIFAR research reactor have been added.

2. GENERAL DESCRIPTION

The BURNMAC module performs four main functions.

Firstly, for initial runs only, the geometry description entered by the user is written as a geometry data pool on FORTRAN unit 4. This description includes the layout of fuel element positions in the reactor core and the division of the positions into burnup zones in which the flux is assumed to be constant in determining burnup. For calculations in which fuel elements are not explicitly represented, the user should consider each required burnup zone to be a 'fuel element'. Normally, this geometry data pool would be used by the flux calculation module. If the reflector geometry is complicated the flux module might not use this data pool, but it is still required by BURNMAC.

Secondly, a core loading giving the fuel element in each position and the irradiation of each burnup zone is maintained and updated. Burnup is simply a matter of advancing the irradiation map. This requires a flux distribution on unit 5 in the form of an AUS FLUXA or FLUXB data pool (the last one on the data pool is used) and a macroscopic fission cross section which is obtained by interpolation in the tabulated input cross sections on unit 10.

Thirdly, cross sections for each zone are obtained by linear interpolation in irradiation in the input cross sections on unit 10 and the interpolated cross sections are written on unit 11 for use by a flux calculation module. The cross sections are written in the order of data for each zone for each fuel element position in turn followed by reflector data. The input to the flux module may be modified as the calculation proceeds if that input is available on FORTRAN unit 7.

Lastly, control of the overall calculation is exercised by BURNMAC rather than by an AUS path program. The input to BURNMAC determines when the flux module will be called. The AUS path program should be a simple loop, *e.g.*

```
1  link burnmac(1,2)
   link pow3d(1,3)
   go to 1
```

The calculation then proceeds till the BURNMAC input is exhausted. Here the flux module is the POW3D diffusion code.

The STATUS data pool on unit 9 is used to pass information from one call of the module to the next. Any serious burnup calculation would require many AUS runs and all five data pools (**gm1**, **fl1**, **st1**, **xs1**, **xs2**) on units 4, 5, 9, 10, and 11, respectively, should be saved to assist in restarts. With all data pools saved, restarts require no additional input. The following information is necessary for error recovery. The STATUS data pool is organised as a **prog** entry to begin each reactor program or cycle followed by a number of **time** entries each followed by loading pattern entries and a zone flux entry for that time. (The **time** and loading pattern entries are written by one link to BURNMAC and the flux entry by the next link - after the flux calculation for that time.) The zone flux entry is used only if a flux data pool is not given. Thus, if a calculation is correct to time T in a program , error recovery is possible by using AUSYS to insert an end-of-file before the entry for the next time in that program and by using the zone flux in the first link to BURNMAC.

3. INPUT SPECIFICATION

3.1. Data Layout

The input data are read using the SCAN input routine [2]. The data are given in the form of a number of entries each of which consists of a keyword followed by a string of data items. More than one entry may be given on each input line. The entries are of six types: material specification, geometry description, program identification, burnup directives, fuel movement directives and flux calculation directives. The entries of each type are described in the following subsections. The following conventions are used.

- Information to be reproduced exactly is given in **bold**.
- Items that the user will replace with an actual value are given in *italics*.
- Optional items are given inside [ ].
- Examples and defaults are given in `constant width` type.

3.2. Material Specification

Material specification entries are normally given first in the initial link to BURNMAC. However, additional **fuel** entries may be given immediately after a **program** entry in subsequent links. A **fuel** entry is given for each type of fuel in the reactor.

fuel *name* **loading** f $i_1, i_2, \ldots, i_n$ $b_1, b_2, \ldots, b_n$ [**or** $r_1, r_2, \ldots, r_n$]

or

fuel *name* **density** f $i_1, i_2, \ldots, i_n$ $b_1, b_2, \ldots, b_n$ [**or** $r_1, r_2, \ldots, r_n$]

where *name* is an alphanumeric 4-character label,
either **loading** or **density** must be given,
f is the fuel loading per element (**loading**) or the fuel loading per unit volume (**density**),
n is the number of sets of cross sections on unit 10 for this fuel type,
i_j is the position number on unit 10 of the j^{th} set of cross sections,
b_j is the corresponding value of the user's burnup units,
r_j is the corresponding value of irradiation, and units of f, and r_j are discussed below.
The inclusion of **or** and r_j is optional. They are not required if $b_j = r_j$ or b_j varies linearly with r_j (see **factors** below).

A **refl** entry is given to identify the non-fuel materials required from unit 10.

refl $l_1, l_2, \ldots, l_n$

where n is the number of materials on unit 10 to be passed directly on to unit 11, and
l_j is the position of the j^{th} material on unit 10.

The last optional entry is

factors h u

where h (default 1) is the ratio of actual volume to volume represented in the flux calculation, and
u (default 1) is the ratio of the user's burnup units to power-time.

A value h is specified to allow real power values. Thus, in an XY model, h would be the core height. In an RZ model reflected at the core mid-plane, h would be 2. A value of u is not given if irradiation values r_j are given on the **fuel** entries. It is retained for compatibility with previous usage where it was used to transform from irradiation to the user's preferred units. For example, a value of $\approx -1.27 \times 10^{-6}$ transforms from Wd to grams of ^{235}U remaining. (The value is negative since mass decreases as irradiation increases.)

The units used are largely at the discretion of the user. However, the following system is recommended as a basis. Throughout AUS, macroscopic cross sections are in cm^{-1} and length is in cm. With the power in watts and the time in days (as specified by the **power** and **step** entries below), the natural units for irradiation r_j are watt days (Wd) per fuel loading unit. Thus with f in grams or g cm^{-3}, r_j is in Wd g^{-1} (*i.e.* MWd t^{-1}). Alternatively, f may be fuel elements or fuel elements per cm^3, in which case r_j is in Wd per fuel element.

Conventionally, the first set of cross sections for a fuel type is xenon free and is used only for irradiation values R, for which $R < r_1 < r_2$ (or $R > r_1 > r_2$). Cross sections for values of R in the range $r_1 \leq R < r_2$ (or $r_1 \geq R > r_2$) are obtained by extrapolation using data at r_2 and r_3.

3.3. Geometry

Geometry data are given only once in an initial BURNMAC run. The data specify the intervals and boundary conditions used in the flux calculation, the layout and labeling of fuel element positions, the division of elements into burnup zones, and the layout of reflector materials. Where appropriate, the data entries have the same form as those used in POW3D.

The mesh intervals and boundary conditions are given exactly as in POW3D using **xm**, **ym**, **zm**, **rm**, **sphere** as appropriate, where each entry is of the form

xm d_L , δ_1 , δ_2 , . . . , δ_n , d_R

where d_L, d_R are left and right boundary conditions and δ_i is the i^{th} mesh interval of a set of n intervals.

As in POW3D, **reg** entries are used to specify the reflector material layout and may also be used to specify the layout of fuel element positions. The form is

reg **mx** i_1 , i_2 , . . . , i_I **my** j_1 , j_2 , . . . , j_J **m**(n)

or

reg **mr** i_1 , i_2 , . . . , i_I **mz** j_1 , j_2 , . . . , j_J **m**(n)

or

reg **mx** i_1 , i_2 , . . . , i_I **my** j_1 , j_2 , . . . , j_J **mz** k_1 , k_2 , . . . , k_K **m**(n)

where the i, j and k are interval numbers, n specifies the n^{th} reflector material for $n > 0$, and $|n|$ specifies the $|n|^{th}$ fuel element position for $n < 0$.

Use of **reg** entries becomes tedious for large numbers of fuel element positions. An alternative specification in terms of channels may be used for regular layouts in which the element positions are bounded by a number of planes. The entries are

lx nx $o_1, o_2, \ldots, o_{mx+1}$

ly ny $p_1, p_2, \ldots, p_{my+1}$

lz nz $q_1, q_2, \ldots, q_{mz+1}$

where nx is the number of channels in the X or R direction, the intervals k satisfying $o_i \le k < o_{i+1}$ are in the i^{th} channel in the X or R direction, **ly** is used for the Y direction (or the Z direction in RZ geometry), and **lz** is for the Z direction in XYZ geometry. The fuel element positions are then numbered (on a printed page) from left to right and top to bottom for each XY channel which does not contain reflector material. In 3D geometry, each fuel element would normally extend over the full core height.
Example:

```
reg   mx  1(1)8   my  1(1)8   m(1)
reg   mx  1(1)6   my  1(1)6   m(0)       ... to clear
reg   mx  5 6     my  5 6     m(1)
lx 3  1 3 5 7   ly 3  1 3 5 7
```

The result is a definition of eight fuel element positions, each with a 2×2 mesh:

```
.  .  .  .  .  .  .  .
.  .  .  .  .  .  .  .
1  1  2  2  .  .  .  .
1  1  2  2  .  .  .  .
3  3  4  4  5  5  .  .
3  3  4  4  5  5  .  .
6  6  7  7  8  8  .  .
6  6  7  7  8  8  .  .
```

The fuel element positions may be given labels by which they are referenced in fuel movement directives. This is done using

positions $pos_1, pos_2, \ldots, pos_{np}$

where pos_i is a 4-character alphanumeric label for the i^{th} fuel element position, and
np must equal the number of fuel element positions.

The first character of each pos_i must be alphabetic. The default labels are the characters `p1, p2, ..., pn`.

In calculations without explicit representation of fuel elements the user should consider burnup zones to be 'elements'. Where fuel elements are explicit, the default option is one burnup zone per element. Division of an element into burnup zones is obtained using

zonereg nx ny nz

where nx is the number of spatial mesh intervals per element in the X direction,
ny similarly specifies the second (Y or Z) direction, and
nz similarly specifies the third (Z) direction.

The **zonereg** entry is followed by a second set of **reg** entries which specify the division of a fuel element into burnup zones. In these **reg** entries, the number of interval numbers following **mx** is nx, the other directions are similar, and n specifies the zone number for the set of mesh intervals instead of a material for standard POW3D **reg** entries. The maximum value of n determines the number of burnup zones per element. The division of an element into zones is usually required only in 3D calculations and printed output is produced on the expectation that the zones are axial zones numbered from bottom to top.

The above entries completely describe the geometry. The initial loading of fuel elements is performed using the fuel movement directive **refuel** described below.

3.4. Program Identification

To identify the current reactor operating program or cycle, the directive used is

program *nprog*

where *nprog* is an integer which identifies the reactor operating program of following directives.

This simply provides a means of labeling an operating program. It causes a **prog** entry to be written on the STATUS data set to assist restarts from that program. The burnup time is reset to zero.

3.5. Burnup Directives

Burnup is specified using two directives. The first is

power p [$fname_1$, $fname_2$ ...]

where p is the power level (normally in watts), and $fname_j$ are an optional set of fuel type names giving the fuel types to be included in the power normalisation. The default is all fuel types. The second directive is

step Δt n

where Δt is the time step (normally in days), and
n (default 1) is the number of steps of length Δt.

The **step** directive causes burnup to take place. For $n > 1$, the power map is recalculated at the beginning of each step using the updated fission cross section for each zone.

3.6. Fuel Movement Directives

The module provides facilities for loading fuel elements, changing their position in the core and also saving elements for later reloading. The loading directive is

refuel n [**delete**]

where n is the number of fuel elements to be loaded, and
delete is given if unloaded fuel elements need not be saved for reloading.

This directive is followed by n **load** or **reload** directives which specify a fuel element to be loaded in the nominated position.

load *pos type num* v_1 , v_2 , . . . , v_{nz}

reload *pos type num*

where *pos* is the element position label,
type is the fuel type which must be identical with one of the values '*name*' on a **fuel** entry,
num is a 4-character alphanumeric fuel element number,
v_i is the irradiation value of the i^{th} zone of the element,
nz is the number of zones per element.

If only one irradiation value is given, it is used for all zones. If no irradiation value is given, the default is the r_1 value for that fuel type. If no element number is given (and hence no irradiations can be given), the default is one more than the last number generated by the module.

The position of fuel elements in the core may be completely respecified using

rearrange pos_1 , pos_2 , . . . , pos_n

or

rearrange no typ_1 num_1 , typ_2 num_2 , . . . , typ_n num_n

where pos_i is a position label giving the location of the element to be moved to the i^{th} element position,
n is the number of elements in the core,
typ_i and num_i specify the type and number of an element in the core to be moved to the i^{th} element position.

Alternatively, movements from one element position to another may be specified using

move $(i_1 , j_1), (i_2 , j_2), \ldots, (i_m , j_m)$

which causes the elements in positions i_1 to i_m to be moved to positions j_1 to j_m respectively.

3.7. Flux Calculation

The directive used to initiate a flux calculation by the module referenced in the AUS path program is simply

flux

Since it may be necessary to change the input to the flux module as the calculation proceeds, the following facility is provided.

replace nl
...

where ... represents nl lines of input which are used to replace existing lines in the input data for the flux module, and columns 73 to 80 must be identical in an existing line and the replacement line.

3.8. HIFAR Specific Directives

The following directives are specific to HIFAR. They may be used to modify the POW3D input data in the standard model of HIFAR with 30×30 mesh intervals [3]. They depend on a particular layout of data in that model.

corerigs ncr lab_1 c_1 , lab_2 c_2 , . . . , lab_{ncr} c_{ncr}

reflrigs nrr lab_1 c_1 , lab_2 c_2 , . . . , lab_{nrr} c_{nrr}

where ncr is the number of rigs in the reactor core,
nrr is the number of rigs in the reflector,
lab_i is the 4-character label of the facility in which the i^{th} rig is to be included,
c_i is the concentration of thermal poison used to represent the i^{th} rig.

The two directives function differently in that **corerigs** gives the complete set of rigs to be included in the core whereas **reflrigs** gives changes to be made to the current state of the reflector.

Additionally, the Coarse Control Arm (CCA) angle may be set in the 3D model using:

cca *angle*

where *angle* is the CCA angle in degrees.

4. REFERENCES

[1]Robinson, G.S. [1986] - CHAR and BURNMAC - burnup modules of the AUS neutronics code system. AAEC/E624.

[2]Bennett, N.W., & Pollard, J.P. [1967] - SCAN - A free input subroutine for the IBM360. AAEC/TM399.

[3]Storr, G.J. [1989] - HIFUME - a fuel management code for HIFAR. ANSTO/NTP/TN132.

AUSED - MODULE FOR EDITING AUS CROSS SECTION DATA POOLS

1. INTRODUCTION

The AUS module AUSED loads, edits, or lists AUS cross section data files. It may be used to maintain basic libraries or to make temporary modifications to neutron cross sections in the user's cross section data files. This module is also used to perform one specialised task, in the process of generating an AUS general purpose library. Namely AUSED converts an AUS file produced by NJOYAUS [Robinson 1993], in which resonance group cross sections are tabulated against potential scatter due to hydrogen, into a data file with subgroup parameters for the resonance theory of MIRANDA.

This report replaces the previous edition by Harrington [1976].

2. CONVENTIONS

The following conventions are adopted in this report:

- module names are given in UPPER CASE,
- types of data file are given in UPPER CASE,
- variable names are given in UPPER CASE,
- information to be reproduced exactly is given in **bold**,
- items that the user will replace with an actual value are given in *italics*,
- omission of some data is indicated by ... ,
- examples and defaults are given in `constant width type`,
- optional items are given inside [ ].

aused: 2

3. FREE STYLE INPUT DATA

Input data are entered in free format style with lower case keywords indicating the data type, *e.g.*

```
sp 0.8  0.2  0.  0.
```

where the keyword **sp** designates that fission spectrum data follow. The data are read using the set of subroutines SKAN [Pollard 1978] which are also very briefly described in Appendix B of the POW3D user's manual [Harrington 1996]. The data features are listed below and apply for both integer and floating point quantities except where otherwise stated.

DATA FEATURES	COMMENTS
Record columns	Data are entered in columns 1 to 72 only. (Each data record is listed when read but `$opt 1,72,0$` anywhere in the data beyond **prelude ... end** may be used to turn off the listing option).
Keywords	May be up to 8 alphanumeric characters but to avoid confusion zero is never used, *e.g.* the last character in **accfo** is the letter o.
Data	Floating point data may be given in abbreviated form, *e.g.* `1  1.  1.-0  1.e+0  1.e 0  1.d-0` are all equivalent. Alphanumeric data may consist of up to 8 alphanumeric characters, starting with an alphabetic character, and must not form keywords.
Extent of data	Different keywords require different extents of data to trail them, *e.g.* for a 4-group library the fission spectrum, **sp**, would require 4 numbers to follow.
Data repeats	`n*d` means that the data entry `d` is repeated `n` times *e.g.* `4*0.` is the same as `0. 0. 0. 0.`

Data increments	`f(i)t` means start from `f` and go to `t` in increments of `i` *e.g.*
	`1(2)7` is the same as `1 3 5 7`
	The same mode must be used for `f`, `i` and `t`, *e.g.*
	`1.(0.5)3.` but not `1(0.5)3`

Readability	May be enhanced by use of special characters, *e.g.* `m(1)=0.07` is equivalent to `m 1 0.07` .
Comment records	A record with an * in column 1 is a comment and is not processed by SKAN. Note that records with ** in columns 1 and 2 are treated as comment data for the current data file (see Section 7.1.4).
Appended comments	Data following ## on an input record is a comment and is ignored, *e.g.* `sp=0.7,0.3,2*0 ## fission spectrum.`

4. INPUT LAYOUT

Examples of UNIX shell scripts and data required for AUSED runs are tabulated below. By default the AUSED data read using Fortran unit 1 is included after *dd2.

Comment	Input
AUS invocation	`aus jobname xs1=/dir/xsdata << 'eof'`

AUS data	`step *`
	`*dd1`
	`  link aused`
	`  end`
	`stop`
	`*dd2`

AUSED data	`prelude ...`
	`start ...`
	`...`
	`end`
	`stop`

AUS termination	`eof`

5. PRELUDE DATA

Prelude data are used to enter the dimensions of variably dimensioned arrays and must precede all other AUSED data. Default values are listed in the Table below. If the default values are satisfactory **prelude** data may be omitted and an AUSED run would then begin with the keyword **start** (see Section 6.1). Any of the default values may be changed by supplying data between the keywords **prelude** and **end**, *e.g.*

```
prelude maxg=55 mscatt=4 maxp=1 print end
```

prelude	Default	Description
maxg=4	201	Maximum number of energy groups
maxp=2	2	Maximum number of P_0, P_1,...,P_{n-1} weighted scattering matrices
mscatt=1	1	Maximum number of scattering temperatures
mscatp=1	1	Maximum number of scattering tabulations against potential scattering
mxsd=1	1	Maximum number of cross section temperatures
mxsdp=1	11	Maximum number of cross section tabulations against potential scattering
mlay=10	120	Maximum number of input materials (including all materials on the input library plus those in the input stream)
mreac=10	11	Maximum number of reactions
maxlv=2	201	Maximum length of a scattering vector
print		Causes the printing of array dimensions
end		Terminating word of **prelude** data

Note: Except for **mlay** the maximum is determined by considering only those materials being changed or entered in the input stream (i.e. those with $layout_i = 2$ or 3; see Section 6.2).

6. AN AUSED RUN

An AUSED run consists of all the data between and including the keywords **start** and **end**. In this section only 'task' data are described, that is, data defining input and output units, error conditions for terminating a run, and indicators regarding which materials are to be modified, copied or loaded from the input stream. In addition to task data an AUSED run may include data for inclusion on an AUS cross-section file as described in Section 7. A run may also include subgroup data which are used for conversion from tabulated resonance integrals to subgroup parameters as described in Section 8. Options to control AUSED output are described in Section 9. Most variables in AUSED have default values and hence are optional.

6.1 START and END of an AUSED Run

A run begins with the keyword **start** followed by input and output units and error conditions.

start $lib_1, lib_2, err_1, err_2$
where lib_1 is the input library Fortran unit number or 0 for initial library loading,
lib_2 is the output Fortran unit number, and if
= 2, output is in a FORMAT suitable for re-input to AUSED with data for each pseudo file ending with `unit=1` (see Section 6.3),
= 6, printed output is produced,
> 6, output is a binary library and lib_1=lib_2 is permitted,
err_1 = 1 the run is terminated if an error occurred in the previous run (default 0),
err_2 = 1 and an error occurs then the current run is terminated before writing an AUS file on Fortran unit lib_2 (default 0).

By default, Fortran units 10, 11, 12 correspond to the cross section data files XS1, XS2, XS3 respectively (Section 4). Fortran unit 2 is normally connected to symbolic file dd12.

A run ends with the keyword **end**.

6.2 LAYOUT of Cross Section Files

The loading, deleting, merging and modification of material data is controlled as follows:

layout $layout_1, layout_2, \ldots, layout_n$
where n $\leq mlay$, and $layout_i$ =1 for i > n,

$layout_i$ {
0 the i^{th} material on lib_1 is deleted for output purposes,
1 the i^{th} material on lib_1 is required unchanged for output,
2 the i^{th} material on lib_1 is modified for output,
3 the i^{th} material data are entered in the input stream,
-1 for lib_2=2 or 6, only pseudo file 1 (Section 7.1) output is produced.

6.3 Input Stream Data Unit

Input stream data for an AUS cross section file are usually read from the standard Fortran input unit, which is 1 in the AUS scheme. The reading of the data can be switched to another unit as follows:

unit *iunit*
where *iunit* is the Fortran unit with input stream data (default value is 1) and may be entered anywhere in a run subsequent to **prelude** data.

7. AUS CROSS SECTION FILE DATA

There are a number of different types of AUS cross section data files with different representations of resonance group data. The general purpose cross section library has subgroup representation of resonances and is used by the data preparation module MIRANDA to prepare cross section data files for particular neutronics calculations. A detailed description of the various types of cross section files is provided in Appendix A of the guide to AUS. The brief description of cross section file data which follows should be sufficient for AUSED use.

An AUS cross section data file or library with data for NN materials consists of NN+1 pseudo files all in a single sequential unformatted data file. Pseudo file 1 consists of library heading records, an index of materials in the library, neutron (and possibly photon) group boundaries and velocities. Each material pseudo file contains burnup and mass information, a fission spectrum, neutron (and possibly photon) cross section data and scattering matrices. Although an AUS cross section data file may include both neutron and photon data, limitations in AUSED allow only neutron data to be modified or loaded onto an AUS library. Nevertheless if AUSED is used to copy or modify a material pseudo file from a coupled neutron/photon data file then it faithfully copies all photon data from the pseudo file.

7.1 PSEUDO FILE 1 DATA

Data for pseudo file 1 are optional, except that for a library which is loaded completely from the input stream, the number of groups must be specified (Section 7.1.1).

7.1.1 Heading Records

Data for the first library heading record of 20 words:

heading *name description*
where **heading** is entered in columns 1 to 7,
name is the library identifier, in columns 9 to 16, (default - `aused`), and
description of the data pool is entered in columns 17 to 72.

Data for the second library heading record of 10 words:

parameters $ipara_1$, $ipara_2, \ldots, ipara_{10}$
where $ipara_1$ is added to the previous library update number (default value 1),
$ipara_2$ is the number of materials on the output library,
$ipara_3$ number of neutron groups (NG) + 1000 × number of photon groups (NGGA),
$ipara_4$ maximum number of reactions,
$ipara_5$ maximum number of cross section temperatures,
$ipara_6$ maximum number of potential scattering values for resonance tabulations of cross sections,
$ipara_7$ maximum number of scattering temperatures,
$ipara_8$ maximum number of potential scattering values for scattering data,
$ipara_9$ maximum number of $P_0, P_1, \ldots, P_{n-1}$ weighted scattering matrices, and
$ipara_{10}$ 100 × NK + INDRES, where NK is the number of kerma factors and INDRES indicates type of tabular data supplied as a function of σ_p for resonance groups (1, 2, 3 or 4 with default type 1) .

Note that for a library which is not loaded entirely from the input stream, AUSED deduces the above parameters by considering all materials including materials to be deleted.

aused: 6

7.1.2 Group Boundaries

Neutron group lethargy and energy boundaries may be entered using either of the following:

lethargy $u_1, u_2, \ldots, u_{ng+1}$
where u_i is the i^{th} lethargy boundary (in ascending order of lethargies), and
ng is the number of groups.

or alternatively,

energy $e_1, e_2, \ldots, e_{ng+1}$
where e_i is the i^{th} energy boundary in eV (in descending order of energies).

7.1.3 Group Velocities

Group velocities may be entered as follows.

velocity $vel_1, vel_2, \ldots, vel_{ng}$
where vel_i is the velocity (cm x $10^8 s^{-1}$) for group i.

7.1.4 Comments

Records with ** in columns 1 and 2 are treated as AUS cross section library comments and are scanned for data up to column 80. File 1 comments may be entered before the first **data** keyword. Comments may be included for each material after the corresponding **data** input.

If a number of comment records *nc* are required from the input library for a particular pseudo data file then *nc* must be entered before the input stream comments for that file thus:

ncomts *nc*
where *nc* is the number of comment records taken from the input library (default *nc*=0); these are followed by comment records from the input stream.

If necessary, *nc*, must be reset for the next pseudo file. If no comments are entered for a particular pseudo file, then all the library comments for that file are used unchanged.

7.2 MATERIAL PSEUDO FILE

Data for a material follows the keyword **data** and are terminated either by the next **data** or **end**. Except for the keyword **data** all material data are optional. Material data must be entered for each $layout_i$ (Section 6.2) with a value of 2 or 3 in the order required for the updated library. For $layout_i$ =2, material data are first read from the pseudo data file on the input library and then modified using input stream data. Where data are not entered or an incomplete list follows any keyword, the data from the input library are used. For material comments see Section 7.1.4.

7.2.1 DATA - Beginning of Input Stream Material Data

Data for a material from the input stream begins with 9 entries,

data *mat*, *source*, *mod*, *ndnar*, *nxst*, *nsp*, *nscatt*, *nscsp*, *np*
where *mat* material name (up to 8 characters) *e.g.* `al`,
source data source (up to 8 characters), e.g. `endfb6`,
mod extent of update (up to 4 characters), e.g. `mod1`,
ndnar = ND+100 × NAR, where NAR is the number of additional reactions and if several materials have the same name, the highest ND is recommended data,

nxst number of neutron cross section temperatures,
nsp number of σ_p values with neutron cross section tabulations,
nscatt number of neutron scatter matrix temperatures,
nscsp number of σ_p values with neutron scatter matrix tabulations,
np number of $P_0, P_1, \ldots, P_{np-1}$ weighted scattering matrices.

For $layout_n = 3$ the default values are:

```
matt0n  aused  orig  1  1  1  1  1  1
```

or part thereof if an incomplete list of data follows the keyword **data**.

7.2.2 BURNUP and Mass Information

The burnup and mass information record consists of 20 words and is described in detail in Appendix A of the AUS guide. Here only word 17 of the burnup record (if negative) is described because it is used when generating a library with subgroup parameters (Section 8). Burnup and mass data are entered as follows.

burnup $name_1, name_2, name_3, burn_7, \ldots, burn_{20}$
where $name_i$ is a nuclide name (up to 8 characters), and if
$burn_{17} < 0$, then $-\bar{D}$ is given, where $\bar{D}$ is the average spacing between resonances (eV).

7.2.3 SP - Fission Spectrum

The fission spectrum may be entered with

sp $sp_1, sp_2, \ldots, sp_{ng}$
where sp_i are terms of the fission spectrum normalised to unit sum.

7.2.4 Temperature and Potential Scatter

Temperature and potential scatter data may be entered as indicated in the table below.

Keyword and Data	Description	Default
xsdtemp $xst_1, xst_2, \ldots, xst_{nxst}$	xst_i are temperatures (K) at which cross sections are tabulated.	all 300
xsdsigp $xsp_1, xsp_2, \ldots, xsp_{nsp}$	xsp_i are σ_p values (σ_{tot} for type 2 data) at which cross sections are tabulated.	all 1.e+20
scattemp $scatt_1, scatt_2, \ldots, scatt_{nscatt}$	$scatt_i$ are temperatures at which scattering matrices are tabulated.	all 300
scatsigp $scatp_1, scatp_2, \ldots, scatp_{nscatp}$	$scatp_i$ are values of potential scatter at which scatter matrices are tabulated.	all 1.e+20

7.2.5 Cross Section Data

Cross section data on an AUS library consist of cross section records and also, for a type 3 library, subgroup parameter records. In a type 3 library, for a group with *nxst* x *nsp* records, the first *nxst* are cross section records and the rest are subgroup parameter records. Using the notation of Appendix A of the AUS guide, both cross section and subgroup parameter records are of the form:

LPS,LV,(XS(K),K=1,LV)

where LV $\begin{cases} = nreac \text{ for all cross section records of a material,} \\ = 3 \text{ for subgroup parameter records of non-fissile materials, and} \\ = 5 \text{ for subgroup parameter records of fissile material.} \end{cases}$

aused: 8

Subgroup parameter records include the following data:

XS(1) is the subgroup weight for absorption,
XS(2) is the subgroup weight for resonance scattering,
XS(3) is the subgroup weight for fission,
XS(4) is the subgroup weight for fission emission, and
XS(5) is the subgroup weight for the ratio of group flux to asymptotic group flux.

For a non-fissile material, XS(3) and XS(4) may be omitted so that XS(3) becomes the subgroup weight for the flux ratio.

Cross section input requirements below are given in terms of the indices of the array in which AUSED stores the data, *eg.* $\sigma(i,j,k,m)$,

where i is the group number,
j is the reaction number,
k is the temperature index, and
m is the potential scatter index.

Cross section input data follow the keyword **xsd** which must always be preceded by **group** or **reac**. The keyword **xsd** may be followed by cross sections for a number of values of temperature and potential scatter. The data may be entered or modified by group:

group *idgp f idtemp idsp* **xsd** $x_1, x_2, \ldots, x_{nr}$

or by reaction

reac *idreac f idtemp idsp* **xsd** $x_1, x_2, \ldots, x_{mng}$

where *idgp* is the group number (default 1),
idreac is the reaction number (default 1),
idtemp is the temperature index (default 1),
idsp is the potential scatter index (default 1),
mng $\leq ng$ x *mreac*,
nr $\leq nreac$ x *maxg*,
f is used to modify the cross section array σ(i,j,k,m) as below (default f=0 and for materials with $layout_i$=3 *f* is set to zero),
x_i is a cross section, subgroup parameter or modification factor.

The following cross-section modification procedure is adopted where the i subscript has been dropped. A prime denotes modified data.

If x=0, then the cross-section is unaltered, *i.e.* $\sigma' = \sigma$,

otherwise x is adjusted, $x' = \begin{cases} x, & \text{for } |x| > 10^{-30} \\ 0, & \text{for } |x| \leq 10^{-30} \end{cases}$

and used to modify the cross-section data, $\sigma' = \begin{cases} |f|\sigma + x', & \text{for } f \leq 0 \\ f\sigma x', & \text{for } f > 0. \end{cases}$

For data entered by group, the x array corresponds to the cross section array in the following order:

(((σ(idgp,j,k,m), j=1,nreac), k=idtemp,nxst), m=idsp,nsp)

and for data entered by reaction in the order

(((σ(i,idreac,k,m), i=1,ng), k=idtemp,nxst), m=idsp,nsp).

If the number of reactions in the input stream data differs from the number on the material data file, the following data must be entered:

nreac *nreac*
where *nreac* is the number of reactions included in the input stream material data.

For a library loaded from the input stream, the default value for *nreac* is $ipara_4$ (Section 7.1.1) and, if still zero, *mreac* (Section 5). Since *nreac* also determines the number of reactions on the output library, it may need to be reset following cross section data entered in the input stream.

For a type 3 data pool with a fissile material entered in the input stream, the keyword

fissile

should be used to indicate that there are subgroup parameters for fission and fission emission.

7.2.6 Scattering Data

Scattering data may be entered as follows:

group *idgp f idtemp idsp idp* **scat** *lss lv* $x_1, x_2, \ldots, x_{lv}$
where *idgp* the group number,
idtemp position in scattering temperature array,
idsp position in potential scatter array,
idp order of P_n weighting (*idp*=n+1),
lss position of self-scatter,
lv length of scattering vector,
x_i vector of out-scatters from the group, and
f, x_i are used in the same way as for cross sections (Section 7.2.5).

Note that for $f > 0$, *lss* and *lv* must have the same values in the input data as in the library data.

8. SUBGROUP PARAMETER FIT

The AUSED program is used to convert an AUS type 1 cross section data file created by NJOYAUS into a type 3 general purpose AUS cross section library. More specifically, AUSED takes group resonance integrals per unit lethargy tabulated against potential scattering by hydrogen and fits subgroup parameters to them. The group scattering matrices are also interpolated so that they are tabulated at fewer values of potential scattering. Input data for fitting subgroup parameters include the following:

- an AUS type 1 cross section file created by NJOYAUS in which the cross section and scatter data for resonance groups are dependent on potential scattering of hydrogen and on temperature,
- two (or four) **group** parameters, from the LAMPOS [Robinson 1993] program, for each resonance group,
- the keywords **sigtfit**, **scatfit** and associated data,
- average resonance spacing $\bar{D}$ may be entered by setting word 17 of the burnup record to $-\bar{D}$,
- the keywords **fissile**, **limres**, **nirc** and associated data, as required.

In the following description the notation follows that of the MIRANDA report [Robinson 1976].

aused: 10

8.1 Subgroup Data

The following data are necessary for each material, and must follow **data** for that material:

sigtfit $sgtft_1$, $sgtft_2$,..., $sgtft_{nsigt}$

where $sgtft_k$ is the total resonance cross section array, σ_k, for use in the subgroup parameter fit of the resonance material. Negative values of σ are permissible and for resonance scatterers they are necessary.

For each group there is a choice between narrow resonance theory, wide resonance treatment or the Hill-Schaeffer [1962] method to set the value of λ used in the fitting of subgroup parameters. (For hydrogen λ=1.) The choice is indicated by **group** data as indicated below. The parameters $group_1$, $group_2$ are described in detail in Section 8.2. The $group_3$ and $group_4$ are optional data which, if supplied, are included in the AUS output file.

group g $group_1$ $group_2$ [$group_3$ $group_4$]

where g is the group number,

$group_1$, $group_2$
- $\leq 0, = 0$ narrow resonance theory is used to set λ,
- $> 0, = 0$ very broad resonance theory is used to set λ,
- $> 0, > 0$ Hill-Schaeffer λ method is used,

$group_3$ average position within the group of absorption resonances,

$group_4$ average position within the group of fission resonances.

For a fissile material, the keyword

fissile

must follow **data** and precede **sigtfit**. Subgroup parameters are then fitted to the fission reaction and those for fission emission are derived by multiplying the fission subgroup parameters for each group by ν, where $\nu = XS_N(3)/XS_N(6)$. The XS_N refers to data from the file created by NJOYAUS even though these data are not standard AUS cross sections.

To indicate that groups with group numbers greater than *limres* (default value *maxg*) are to be treated as non-resonance groups, the following data should be used:

limres *limres*

Scattering matrices, which on the input library are tabulated against σ_{Hi}, may be retabulated against an arbitrary set of $\hat{\sigma}_p$ (Section 8.2) on the output library, as follows:

scatfit $sctft_1$,$sctft_2$,...,$sctft_n$

where $n \leq mscatp$,and

$sctft_i$ is an effective scattering cross section $\hat{\sigma}_{pi}$.

The **nirc** directive distinguishes the groups for which resonance integrals have been computed as a sum of J functions from those for which resonance integrals have been derived from a numerical solution of the slowing down equations. Resonance integrals derived from the slowing down equations include a non-isolated resonance correction which is removed by AUSED before fitting subgroup parameters if word 17 of **burnup** data has been set to -$\bar{D}$. However, high energy group data do not have resonance correction factors and in these groups a simple approximation for the flux depression is used. The following data may be used to stop removal of the non-isolated resonance correction for all groups preceding group *nirc*.

nirc *nirc*

where for groups with higher energies than for group number *nirc* a simple approximation is used for the group flux depression, whereas
for group *nirc* and lower energy groups if $\bar{D}$ is given then the non-isolated resonance correction is removed and the group flux depression, derived from the slowing down calculation in GENRI [Robinson 1993], is taken from the input library.

8.2 Fitting of Subgroup Parameters

The following description should be used in conjunction with the MIRANDA report [Robinson 1976]. In this section the index i indicates each tabulated value of the resonance integrals while index k represents subgroups. For each resonance group, resonance integrals per unit lethargy, I′, for absorption, total, scattering and possibly fission are extracted from an AUS file created by NJOYAUS as follows.

$$I'_a(\sigma_{Hi}) = XS_N(6, \sigma_{Hi}) + XS_N(7, \sigma_{Hi}) \quad \text{- absorption}$$
$$I'_{tot}(\sigma_{Hi}) = XS_N(5, \sigma_{Hi}) - \sigma_p - \sigma_{inel} \quad \text{- total}$$
$$I'_s(\sigma_{Hi}) = I'_{tot}(\sigma_{Hi}) - I'_a(\sigma_{Hi}) \quad \text{- scattering}$$
$$I'_f(\sigma_{Hi}) = XS_N(6, \sigma_{Hi}) \quad \text{- fission, for a **fissile** material only}$$

where σ_{Hi} is the potential scattering due to hydrogen,
σ_p is the potential scattering due to the resonance material , and
σ_{inel} is given by $-group_1$ if $group_1$ is negative otherwise it is zero.

The other quantity to be fitted is the ratio, I_d', of group flux to asymptotic group flux (i.e. group flux depression). This ratio is derived for group g as follows:

for $g < nirc$, $I'_d(\sigma_{Hi}) = 1 - I'_{tot}/(\sigma_p + \sigma_{Hi})$

for $g \geq nirc$, $I'_d(\sigma_{Hi}) = XS_N(NAR + NK + 7) \times XS_N(NAR + NK + 8)/\phi_{0i}\tau$

where $\phi_{0i} = 1/(\xi \times \sigma_p + \sigma_{Hi})$

$$\xi = 1 + \frac{\alpha \log \alpha}{1 - \alpha}$$

$$\alpha = \left(\frac{A - A_0}{A + A_0}\right)^2$$

A = atomic mass,
$A_0 = 1.008665$,
τ is the group lethargy width,
XS_N are AUS file data produced by NJOYAUS not standard cross sections,
NAR is the number of additional reactions on the AUS file, and
NK is the number of kerma factors on the AUS file.

For each temperature the resonance integral is regarded as a function of effective scattering cross section $\hat{\sigma}_{pi}$. The value of $\hat{\sigma}_{pi}$ is given by

$$\hat{\sigma}_{pi} = (\sigma_{Hi} + \lambda_i \sigma_p) \frac{I_a'(\infty) + I_s'(\infty)}{I_a'(\infty) + \lambda_i I_s'(\infty)}, \quad \text{if } I_s'(\infty) \geq 0, \text{ or}$$

$$\hat{\sigma}_{pi} = \sigma_{Hi} + \lambda_i \sigma_p, \quad \text{if } I_s'(\infty) < 0 \text{ and absorption is predominant, or}$$

$$\hat{\sigma}_{pi} = \frac{\sigma_{Hi} + \lambda_i \sigma_p}{\lambda_i}, \quad \text{if } I_s'(\infty) < 0 \text{ and resonance scatter is predominant,}$$

where $I_a'(\infty)$, $I_s'(\infty)$ are the absorption and resonance scattering integrals at infinite dilution, λ_i values are determined in each group either by (1) narrow resonance theory, or (2) wide resonance treatment or (3) the Hill-Schaeffer method. The choice is indicated by the input parameters $group_1$, $group_2$ as described below.

(1) $group_1 \leq 0$, $group_2 = 0$ implies narrow-resonance theory and the setting of λ_i=1 for the material.
If $group_1 < 0$ then $group_1 = -\sigma_{inel}$, and inelastic scattering is used in the derivation of $I'_{tot}(\sigma_H)$ as above.

(2) $group_1 > 0$, $group_2 = 0$ implies a very wide resonance extending over many groups. In this case λ_i is set to

$$\lambda_i = \beta / \beta_H$$

where $\beta = \xi$ for $\alpha > \dfrac{E(g+1)}{E(g)}$,

or

$$\beta = \frac{1 - E(g+1)/E(g) - \alpha\tau}{(1-\alpha)} \quad \text{for } \alpha \leq \frac{E(g+1)}{E(g)},$$

$$\beta_H = 1 - E(g+1)/E(g),$$

E(g), E(g+1) are the energy boundaries of the group.

(3) $group_1 > 0$ $group_2 > 0$ implies the Hill- Schaeffer λ method, with

$$group_1 = \bar{\sigma}_0, \quad group_2 = \overline{\left(\frac{2E_r}{\Gamma}\right)}$$

A form of the Hill-Schaeffer method is used to set λ_i for the material as follows:

$$\lambda_i = 1 - \frac{2\alpha C_i}{x} \tan^{-1}\left(\frac{x}{C_i(1+\alpha)}\right)$$

where $C_i = \sqrt{1 + \dfrac{\bar{\sigma}_0}{\sigma_{Hi} + \sigma_p}}$

$$x = \overline{\left(\frac{2E_r}{\Gamma}\right)}(1-\alpha)$$

σ_0 is the peak microscopic total resonance cross section,
E_r is the resonance energy,
Γ is the total level width of the resonance,
bar denotes some reasonable estimate of the group average values.

For materials with mass less than 200, a check of each resonance group is made to see whether it should be changed to a non-resonance group. Subgroup parameters are not fitted if for a particular group the absorption integral for each value of σ_{Hi} for each temperature is given to within 1% by the formula

$$I'_a(\infty) \times \frac{\hat{\sigma}_p}{(\hat{\sigma}_p + I'_{tot}(\infty))}$$

where $\hat{\sigma}_p$ is calculated using λ_i for a very wide resonance.

AUSED removes the MIRANDA type [Robinson 1976] non-isolated resonance correction before fitting subgroup parameters if the average resonance spacing, $\bar{D}$, is given for the material. The correction is later reapplied in MIRANDA. This procedure is necessary to obtain reasonable fits in the keV energy range for the actinides. The procedure is that outlined by equations 5.47 to 5.53 [Robinson, 1976]. A little more detail is given here. The integrals above have been denoted as I′ to indicate integrals including non-isolated resonance effects. The integrals with this effect removed are denoted by I. All integrals including group flux depression are modified using

$$I_x(\sigma_{Hi}) = I'_x(\sigma_{Hi})/\bar{W}(\sigma_{Hi})$$

where $\bar{W}(\sigma_{Hi})$ is calculated for each σ_{Hi} using the following iterative procedure and equations [Robinson 1976].

Three iterations are used with $\bar{W}(\sigma_{Hi})$ initially set to 1. First, I_a and I_s are calculated from the above equation for the current values of $\bar{W}$. These integrals are regarded as functions of $\hat{\sigma}_p$ and at each σ_{Hi} the value of the corresponding narrow resonance integral is obtained by linear interpolation in $\sqrt{\hat{\sigma}_p}$ to give the integrals at σ_p' $(= \sigma_{Hi} + \sigma_p)$. The corresponding absorption and scattering cross sections are then calculated using equations 5.48 and 5.49. The resonance escape probability is calculated using equation 5.50 and a value of $\bar{W}$ is obtained from 5.51. This value is modified using equations 5.52 and 5.53 to allow for the width of the group. This terminates the iterative loop.

Using a modification of the Russian subgroup method [Nikolaev et al. 1971], the absorption, total, fission resonance integrals and the group flux depression (I_{ai}, $I_{ai}+I_{si}$, I_{fi} and I_{di}) can each be approximated by a sum of the form:

$$I_{xi} = \sum_k \frac{w_{xk}\hat{\sigma}_{pi}}{\hat{\sigma}_{pi}+\sigma_k}$$

The subgroup parameters, w_{xk} are fitted by a least squares method (Doherty 1972) to minimise

$$\sum_i \left(1 - \sum_k \frac{w_{xk}\hat{\sigma}_{pi}}{(\hat{\sigma}_{pi}+\sigma_k)\, I_{xi}}\right)^2$$

where $w_{xk} \geq 0$ except when fitting I_{tot} in which case $w_{xk}\sigma_k \geq 0$.

The subgroup parameters written to the file for resonance scattering are the differences between the subgroup parameters fitted to the total and absorption resonance integrals. The cross section records for $\sigma_{Hi} = \infty$ are written on the output file with XS(8, ∞) set to zero. For the first temperature only, the $group_j$ are added to the cross section record as XS(NX+NK+j). For those resonance groups which are changed to non-resonance groups, only one cross section record is written which is the record for $\sigma_{Hi} = \infty$ for the first temperature modified by setting XS(8), XS(NX+NK+1) and XS(NX+NK+2) to zero. The scattering matrix used for these groups is that for the lowest value of σ_{Hi} and for the lowest temperature. For resonance groups, the set of scattering matrices are treated as a function of $\hat{\sigma}_p$ and interpolated to give matrices at the **scatfit** input values using linear interpolation in $\sqrt{\hat{\sigma}_p}$.

9. EDITING OF OUTPUT

In Section 9.1 options are described for selectively reducing output on the printer or Fortran unit 2. The options described in Section 9.2 enable the insertion of cross section or scattering data at values of temperature or potential scatter intermediate to those already on the cross section file.

aused: 14

9.1 Unit 2 or 6 Output

The **print**, **output** and **nuclide** options allow the user to be selective with output on Fortran unit 2 or printed output on unit 6. Fortran unit 2 is normally connected to the symbolic file dd12.

print *ipr*

where $ipr = \begin{cases} 0 \text{ switches off the printing of input stream material data (except for } \textbf{data} \text{ itself),} \\ 1 \text{ switches the printing on.} \end{cases}$

The default is: `print 1`

output $iopt_1, iopt_2, \ldots, iopt_{ng}$

where ng is the number of groups, and for group i, if

$$iopt_i = \begin{cases} 0 \text{ no output is produced} \\ 1 \text{ cross section and scattering data are produced} \\ 2 \text{ only cross section data are produced} \\ 3 \text{ only scattering data are produced} \end{cases}$$

Default values of $iopt_i$ are all 1.

nuclide $nucld_1, nucld_2, \ldots, nucld_8$

where if $nucld_i$ =-1 , no output is produced for the i^{th} item in the following list:

data, burnup, sp, xsdtemp, xsdsigp, scattemp, scatsigp and comments

Default values of $nucld_i$ are all 0.

9.2 Reordering of Cross Section and Scattering Data

Library data may be reordered using **libxt**, **inxt**, **insp**, etc. keywords with integer data. In the table below, **xst**, **xsp**, **scatt** and **scatp** are temperature and potential scatter arrays (Section 7.2.4). Cross section data with a temperature index i on input, for example, has the index $libxt_i$ on output.

Keyword and Data	Nuclear Data	Input tabulated against	Output tabulated against
libxt $libxt_1, libxt_2, \ldots, libxt_{nxst}$	library cross sections	xst(i)	xst($libxt_i$)
inxt $inxt_1, inxt_2, \ldots, inxt_{nxst}$	input stream cross sections	xst(i)	xst($inxt_i$)
libxp $libxp_1, libxp_2, \ldots, libxp_{nsp}$	library cross sections	xsp(i)	xsp($libxp_i$)
inxp $inxp_1, inxp_2, \ldots, inxp_{nsp}$	input stream cross sections	xsp(i)	xsp($inxp_i$)
libst $libst_1, libst_2, \ldots, libst_{nscatt}$	library scattering data	scatt(i)	scatt($libst_i$)
inst $inst_1, inst_2, \ldots, inst_{nscatt}$	input stream scattering data	scatt(i)	scatt($inst_i$)
libsp $libsp_1, libsp_2, \ldots, libsp_{nscsp}$	library scattering data	scatp(i)	scatp($libsp_i$)
insp $insp_1, insp_2, \ldots, insp_{nscsp}$	input stream scattering data	scatp(i)	scatp($insp_i$)

The default values are set at the beginning of each run so that $libxt_i$=i, $inxt_i$=i, etc. for all i. The values may be changed as often as necessary within a run.

As an example, consider including extra cross section data in the input stream at a temperature (say 600 K) that lies between two existing temperature tabulations (say 900 K and 300 K) on a data file. The following data would be used:

```
xsdtemp 900. 600. 300.                    - see section 7.2.4
libxt 1 3
inxt 2
...                                       - cross section data at 600 K
```

10. EXAMPLES

Input data for three AUSED examples follow. In the first example 4-group cross section data for a particular reactor system are entered in the input stream to create an AUS cross section data file, bvh.lib1, which is then listed. In the second example subgroup parameters are fitted to type 1 resonance data. Cross section data for resonance groups are written in a FORMAT suitable for input to AUSED onto Fortran unit 2 which is connected by default to the symbolic file dd12. In the third example the reaction 8 data are multiplied by 0.02 for all groups of the sixteenth library material.

(1) Example of the loading of a 4-group library from the input stream

```
aus xs1=bvh.lib1 << 'eof'
*dd1
step *
      link aused
      end
stop
*dd2
start 0 10 0 layout 4*3
parameters 1 4 4 3 6*1
data fuel pow orig  1       1       1       1       1       1
sp 7.53564E-01 2.46436E-01 0.00000E+00 0.00000E+00
group   1  0  1  1 xsd
 1.54481E-01 4.32998E-04 3.73274E-04
group   1  0  1  1  1 scat
    1    2 6.03198E-02 9.37282E-02
group   2  0  1  1 xsd
 3.68630E-01 2.66749E-03 3.30167E-03
group   2  0  1  1  1 scat
    1    3 3.08395E-01 4.88109E-02 8.75701E-03
group   3  0  1  1 xsd
 9.02355E-01 2.71099E-02 3.96448E-02
group   3  0  1  1  1 scat
    2    3 5.09412E-03 2.21329E-01 6.48822E-01
group   4  0  1  1 xsd
 2.00955E+00 6.50450E-02 9.77554E-02
group   4  0  1  1  1 scat
    3    3 1.66342E-07 4.77318E-02 1.89677E+00
data al   pow orig  1       1       1       1       1       1
sp 7.53564E-01 2.46436E-01 0.00000E+00 0.00000E+00
group   1  0  1  1 xsd
 1.16556E-01 2.83498E-04 0.00000E+00
group   1  0  1  1  1 scat
```

aused: 16

```
    1    2 1.00141E-01 1.61320E-02
group   2  0  1  1 xsd
 1.05169E-01 7.01732E-04 0.00000E+00
group   2  0  1  1  1 scat
    1    2 1.03960E-01 5.06778E-04
group   3  0  1  1 xsd
 9.05011E-02 5.07732E-03 0.00000E+00
group   3  0  1  1  1 scat
    2    3 5.36885E-04 7.20487E-02 1.28382E-02
group   4  0  1  1 xsd
 9.34386E-02 1.21567E-02 0.00000E+00
group   4  0  1  1  1 scat
    2    2 1.27736E-03 8.00045E-02
data c    pow orig  1      1      1      1      1      1
sp 7.53564E-01 2.46436E-01 0.00000E+00 0.00000E+00
group   1  0  1  1 xsd
 1.49824E-01 3.30061E-06 0.00000E+00
group   1  0  1  1  1 scat
    1    2 1.27331E-01 2.24898E-02
group   2  0  1  1 xsd
 3.44423E-01 8.94773E-06 0.00000E+00
group   2  0  1  1  1 scat
    1    3 3.40191E-01 4.22309E-03 4.23284E-08
group   3  0  1  1 xsd
 3.74146E-01 1.05663E-04 0.00000E+00
group   3  0  1  1  1 scat
    2    3 1.58544E-03 2.75876E-01 9.65788E-02
group   4  0  1  1 xsd
 3.63541E-01 2.48671E-04 0.00000E+00
group   4  0  1  1  1 scat
    2    2 8.07923E-03 3.55213E-01
data h2o  pow orig  1      1      1      1      1      1
sp 7.53564E-01 2.46436E-01 0.00000E+00 0.00000E+00
group   1  0  1  1 xsd
 1.70681E-01 2.15702E-04 0.00000E+00
group   1  0  1  1  1 scat
    1    2 4.63863E-02 1.24079E-01
group   2  0  1  1 xsd
 4.42615E-01 8.82911E-04 0.00000E+00
group   2  0  1  1  1 scat
    1    3 3.72542E-01 5.90997E-02 1.00901E-02
group   3  0  1  1 xsd
 1.07516E+00 9.71049E-03 0.00000E+00
group   3  0  1  1  1 scat
    2    3 3.44408E-03 2.41371E-01 8.20634E-01
group   4  0  1  1 xsd
 2.25085E+00 2.28414E-02 0.00000E+00
group   4  0  1  1  1 scat
    3    3 1.78214E-07 5.17295E-02 2.17628E+00
end
start 10 6 1 layout 4*1 end
```

(2) Example of the fitting of subgroup parameters

```
aus dd12=pu241.subg xs1=aus.bvhlib1 xs2=aus.bvhlib2 << 'eof'
step *
*dd1
      link aused
      end
stop
*dd2
prelude maxg=127,mreac=10,maxp=1,mxsdp=23,print,end
start 11 10 1 layout 2
data pu241 jpgym767 mod1
burnup am241 pu242 no0 5 1.69*15 6*0. 3.316e-11 241.056 3*0. 0942410
fissile
sigtfit 0.625 2.5 10. 40. 160. 640. 2560.
group   57 1.37+3 1.31+2
group   59 1.21+3  1.15+2
group   61 5.17+2  1.15+1
group   62 1.61+2  1.03+2
group   63 6.47+2  7.43+1
group   64 2.20+3  6.96+1
end start 10 2 1 nuclide 8*-1
output 56*0 2 0 2 0 4*2 63*0
layout 1
end
eof
```

(3) Example of the modification of cross sections on a user's cross section data pool

```
aus xs3=xssi <<'eof'
*dd1
step *
      link aused(1,6)
      end
stop
*dd6
prelude maxg=31 maxlv=31 maxp=6 end
start 12 12
layout 15*1 2 1
data
* to calculate displacements per atom in silicon where
* si is the 16th material on a 17 material, 31 group library
* multiply si neutron damage energy by .8/(2*20)
reac 8 1 xsd 31*2.0-2
end
```

11. ACKNOWLEDGEMENTS

In an attempt to provide a complete description of the fitting of subgroup parameters some sections have been taken from an unpublished AUS note [Robinson 1993a] and also the MIRANDA report [Robinson, 1976].

aused: 18

12. REFERENCES

Doherty, G. [1972] - Applications of collision probabilities in reactor physics. Ph.D. Thesis, University of New South Wales.

Harrington, B.V. [1976] - AUS module AUSED - an editing program for AUS cross section data pools. AAEC/E389.

Harrington, B.V., Pollard, J.P., Barry, J.M. [1996] - POW3D - Neutron diffusion module of the AUS system, a user's manual. ANSTO/E726.

Hill, J.G. & Schaefer, G.W. [1962] - An improved approximation for the calculation of resonance integrals. English Electric Report W/AT-1035.

Nikolaev, M.N., Ignatov, A.A., Isaev, N.V. & Khohlov, V.F. [1971] - Method of subgroups for accounting of resonance structure of cross sections in neutron calculations, part 2. At. Ehnerg. 30(5)426

Pollard, J.P. [1978] - SKAN - A Free input labelled output variable dimensioning routine for the IBM360 computer. AAEC/E431.

Robinson, G.S. [1976] - AUS module MIRANDA - A data preparation code based on multiregion resonance theory. AAEC/E410.

Robinson, G.S. [1993] - Generation and validation of a cross section library based on ENDF/B-VI for the AUS neutronics code system. ANSTO/E712.

Robinson, G.S. [1993] - The Generation and use of resonance data on an AUS library as at 30 July 1993, unpublished AUSNOTE.