SANDIA REPORT

SAND2003-4173
Unlimited Release
Printed June 2004

ITS Version 5.0: The Integrated TIGER Series of Coupled Electron/Photon Monte Carlo Transport Codes

Brian C. Franke, Ronald P. Kensek, Thomas W. Laub

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2003-4173
Unlimited Release
Printed June 2004

ITS Version 5.0: The Integrated TIGER Series of Coupled Electron/Photon Monte Carlo Transport Codes

Brian C. Franke, Ronald P. Kensek, and Thomas W. Laub
Simulation Technology Research Division

Abstract

ITS is a powerful and user-friendly software package permitting state-of-the-art Monte Carlo solution of linear time-independent coupled electron/photon radiation transport problems, with or without the presence of macroscopic electric and magnetic fields of arbitrary spatial dependence. Our goal has been to simultaneously maximize operational simplicity and physical accuracy. Through a set of preprocessor directives, the user selects one of the many ITS codes. The ease with which the makefile system is applied combines with an input scheme based on order-independent descriptive keywords that makes maximum use of defaults and internal error checking to provide experimentalists and theorists alike with a method for the routine but rigorous solution of sophisticated radiation transport problems. Physical rigor is provided by employing accurate cross sections, sampling distributions, and physical models for describing the production and transport of the electron/photon cascade from 1.0 GeV down to 1.0 keV. The availability of source code permits the more sophisticated user to tailor the codes to specific applications and to extend the capabilities of the codes to more complex applications. Version 5.0, the latest version of ITS, contains (1) improvements to the ITS 3.0 continuous-energy codes, (2) multigroup codes with adjoint transport capabilities, and (3) parallel implementations of all ITS codes. Moreover, the general user friendliness of the software has been enhanced through increased internal error checking and improved code portability.

Acknowledgment

There have been many people involved in the development of the ITS codes over the years. We have undoubtedly lost track of the contributions of some, and even if we had not, there would be too many to list here. We are grateful for the many improvements they have suggested. While recognizing that this is necessarily an abbreviated list, we would like to acknowledge some of the contributions that we deem to have been most signicant or most recent.

The ITS codes owe most of their development to John Halbleib. He wrote the TIGER code and was the primary developer of the ITS codes throughout most of their history. The original TIGER code was built on the ETRAN code developed by Steve Seltzer and Martin Berger, both of NIST (the National Institute of Standards and Technology). Further collaborations with NIST, led to improvements of the algorithms and cross sections used by ITS. Tom Mehlhorn played an important role in the first integration of the Integrated TIGER Series.

This release of ITS includes, for the first time, the multigroup capability. The development of MITS (Multigroup ITS) owes much to the contributions of Jim Morel of Los Alamos National Laboratories. MITS is also dependent on the CEPXS cross section generating code that was principally developed by Len Lorence.

The parallel capability of the ITS codes was implemented by Greg Valdez. Much testing and many improvements have been contributed by Wesley Fan. Recent improvements in the physical models in the ITS codes were implemented by Veronica Klein. Recent improvements in the regression testing of ITS are due to the coverage analysis work of Lisa Cordova.

The conversion of the CEPXS code to Fortran 90 was accomplished by Clif Drumm. The conversion of the XGEN code to Fortran 90 was accomplished by Martin Crawford.

We would also like to acknowledge the recent helpful suggestions from friendly users, Ken Adams of SAIC and Wesley Fan.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Contents

Figures

Tables

Last Modified: March 18, 2004

1 Introduction to ITS

The TIGER series of time-independent coupled electron/photon Monte Carlo transport codes is a group of multimaterial and multidimensional codes designed to provide a state-of-the-art description of the production and transport of the electron/photon cascade. The continuous-energy ITS codes are based primarily on the ETRAN model[1], which combines microscopic photon transport with a macroscopic random walk[2] for electron transport. The multigroup ITS codes are based primarily on the MORSE model[3], with a modification by Sloan[4] to model electron elastic scattering, both of which preserve the angular moments of scattering with discrete scattering angle models. Emphasis is on simplicity of application without sacrificing the rigor or sophistication of the physical model.

1.1 History of the TIGER Series

Table 1 chronicles the development of the TIGER series, beginning with the EZTRAN[5] and EZTRAN2[6] codes in the early 1970's. These codes were basically user oriented versions of the ETRAN codes. They were severely limited in their application to real physical problems because of their restriction to a single homogeneous material. Overcoming this limitation was the original motivation for the development of the TIGER series.

Table 1. Chronology of TIGER series development

Code	Date	Released	Dimension
EZTRAN	Sep 71	Yes	1-D
EZTRAN2	Oct 73	Yes	2-D/3-D[a]
TIGER	Mar 74	Yes	1-D
CYLTRAN	Mar 75	Yes	2-D/3-D[a]
CYLTRANM	Jun 77	No	2-D/3-D[a]
TIGERP	May 78	Yes	1-D
SPHERE	Jun 78	Yes	1-D
ACCEPT	May 79	Yes	3-D
SPHEM	Jul 79	No	1-D/3-D[a]
CYLTRANP	Late 81	No	2-D/3-D[a]
ACCEPTM	Late 81	No	3-D

[a]The first dimension refers to the material geometry, while the second dimension refers to the description of the particle trajectories.

TIGER[7], CYLTRAN[8], and ACCEPT[9] are the base codes of the series and differ primarily in their dimensionality and geometric modeling. TIGER is a one-dimensional multilayer code. CYLTRAN employs a fully three-dimensional description of particle trajectories within an axisymmetric cylindrical material geometry and quite naturally finds application in problems involving electron or photon beam sources. ACCEPT is a general three-dimensional transport code that uses the combinatorial-geometry scheme developed at MAGI[10, 11].

The original base codes were primarily designed for transport from a few tens of MeV down to 1.0 and 10.0 keV for electrons and photons, respectively. Furthermore, fluorescence and Auger processes in the base codes are only allowed for the K-shell of the highest atomic number element in a given material. For some

applications it is desirable to have a more detailed model of the low energy transport. In the TIGERP[12] and CYLTRANP[13] codes, we added the more elaborate ionization/relaxation model from the SANDYL code[14] to the TIGER and CYLTRAN codes, and we extended photon transport down to 1.0 keV (all member codes of the ITS system allow transport over the range 1.0 GeV to 1.0 keV).

In CYLTRANM[15], we combined the collisional transport of CYLTRAN with transport in macroscopic electric and magnetic fields of arbitrary spatial dependence using a Runga-Kutta-Fehlberg algorithm[16] to integrate the Lorentz force equations. An important modification of this algorithm[17] made possible the development of the ACCEPTM code[18] , which combines the collisional transport of the ACCEPT code with macroscopic field transport. SPHERE[19] and SPHEM[20] were two special purpose codes that were restricted to multiple concentric spherical shells without and with macroscopic field transport, respectively.

EZTRAN, EZTRAN2, and SPHEM are considered obsolete. Before ITS, that still left us with eight separate code packages to maintain. Five of these – TIGER, CYLTRAN, ACCEPT, TIGERP and SPHERE – had been publicly released and were disseminated through the Radiation Shielding Information Center at Oak Ridge National Laboratory. CYLTRANM, CYLTRANP and ACCEPTM were not publicly released, but were maintained locally for use throughout Sandia National Laboratories. Maintaining multiple code packages had become quite burdensome for us as well as for users of the codes. As a result, important modifications were not being implemented in a timely fashion. Furthermore, the multiplicity of packages had resulted in uneven development of the various codes such that each code had unique features that had not yet been implemented in the other codes.

In order to remedy this situation we developed ITS (the Integrated TIGER Series), whose full implementation superseded all other versions of the TIGER series codes[21]. The combined program library file was obtained by integrating the eight codes in the first three columns of Table 2 in such a way as to minimize the repetition of coding that is common to two or more of these codes. This process led quite naturally to the development of a new code, ACCEPTP. In ACCEPTP, the improved low-energy physics of the SANDYL code was added to the ACCEPT code. Those individual codes appearing in Table 1, but not in Table 2, were of a more specialized nature than the others and were no longer supported since their function was duplicated by at least one of the ITS codes. Additional cross-section data and associated logic allowed transport from 1.0 keV to 1.0 GeV[22] for both electrons and photons. A new free-format, order-independent input procedure based on descriptive keywords and maximum use of defaults and internal error checking resulted in a very simple and user-friendly input scheme. Integration of the various codes resulted in the availability of additional common options for each code.

Table 2. ITS member codes

Standard Codes	Enhanced Ionization/ Relaxation (P Codes)	Macroscopic Fields (M Codes)	Multigroup (MITS Codes)
TIGER	TIGERP		MITS TIGER
CYLTRAN	CYLTRANP	CYLTRANM	
ACCEPT	ACCEPTP	ACCEPTM	MITS ACCEPT

ITS was further enhanced throughout the 1980's to improve the physical models, enhance user friendliness, and increase the options available to the user[23]. Beginning in 1994, a set of multigroup ITS codes (MITS) was developed[24, 25, 26]. The MITS codes inherited the ITS combinatorial-geometry logic, but through alternative physical models, it added a capability for adjoint transport calculations. In 1997, we began the development of a capability of tracking particles on CAD geometries[27]. Throughout the 1990's the original continuous-energy combinatorial-geometry ITS codes were further enhanced with complicated subzoning capabilities, more source options, and flexible biasing schemes. In 2000, ITS, the MITS codes, and the CAD tracking capability were integrated into a single package of codes. In 2002, ITS 5.0 was in-

ternally released at Sandia with a verified capability of adjoint particle transport on CAD geometries[28]. Table 2 shows the code options available in the current version of ITS.

From top to bottom, the codes grouped by row in Table 2 will be referred to as the TIGER codes, the CYLTRAN codes, and the ACCEPT codes, respectively. From left to right, the codes grouped by column will be referred to as the standard codes, the PCODES, the MCODES, and the MITS codes, respectively. We acknowledge that some confusion may result from a dual context-dependent use of the term "ITS". In general, we will use "ITS" to mean the complement of the MITS codes (i.e., the first three columns of Table 2), and we will use "ITS codes" to refer to the entire series of codes including the MITS codes.

Since the initial release[29], feedback from the user community has been of great benefit to the development of the ITS code system. As a consequence of this feedback, subsequent versions have implemented important improvements in physical accuracy, new capabilities, variance reduction, and user friendliness.

Last Modified: February 16, 2004

2 Overview of the Documentation

The documentation has been written with the intention that it may be used as a reference manual for the expert user and the beginning user alike. Brief overviews are provided on step-by-step execution of the program (in Running ITS), on preprocessor options (in Code Options), and on program keywords (in Summary of ITS Keywords). These may serve as quick references for the expert and as an initial outline for the novice. Each of these are followed by more detailed explanations that in turn may be associated with sections containing further details and/or theory for the code features. It is intended that all users should begin by referring to the Running ITS section and follow references in the documentation as necessary to gain further understanding of features to be used. Users are encouraged to peruse the entire manual to gain a better understanding of the code package and how various options may be employed. However, some sections apply to restricted subsets of the code options available and may not be relevant to the user's specific types of problems.

2.1 Creating the Manual

The ITS manual can be constructed from the LaTeXdocuments contained in the Docs directory.

To obtain a specific version of the manual, a cvs checkout may be used to request that version. One may use the date from the cover of the ITS manual (e.g., cvs checkout -D "April 5, 2002" its), or one may use a release tag (e.g., cvs checkout -r ITSversion5.0 its).

Each section is contained in the Docs/Sections directory as a separate *.tex file. Each of these files can be compiled individually (see the following section). The Docs/Misc directory contains the bibliography file (ITS.bib) that should be used whenever an individual file or the manual is being compiled so that citations can be resolved. All of the figures used by the documents are contained in the Docs/Graphics directory and are encapsulated postscript files. When compiling, these should be in the same directory as the document being compiled.

To compile the manual using a standard LaTeXcompiler, a Makefile is available in the Docs directory. "make" will create the manual in DVI format. "make ps" will create the manual in postscript format. "make pdf" will create the manual in PDF format. "make view" will create the manual and display it using xpdf. "make clean" will remove all of the temporary files used to create the manual. "make realclean" will remove all temporary files and the dvi, ps, and pdf files.

To compile the manual in Scientific WorkPlace®[30]: The directory Docs/Misc contains two scripts for converting the section documents into sections to be included in the manual. The "make_sections" script should be executed. It uses the "comment_headers" script to comment out portions of the section documents. The section documents will be placed in the Docs directory. The Docs directory also contains the ITS.tex document. This is the main document that includes each of the sections and must be compiled to create the manual. The ITS.bib and Graphics files are also necessary for compiling the manual. A copy of each of the section documents must be placed in the directory in which the ITS.tex file is to be compiled and must also be placed in the swp/temp directory. A copy of each of the Graphics files must be placed in the directory in which the ITS.tex file is to be compiled. The ITS.bib file must be placed in the swp/TCITeX/bibtex/bib directory. Open the ITS.tex file with Scientific WorkPlace, and perform a "Typeset Compile" with the "Generate a Bibliography" and "Generate an Index" options selected. This will compile the ITS manual into a DVI file. The ITS.dvi file can be viewed using "Typeset Preview". (To properly format the manual, it is necessary to trick Scientific WorkPlace into thinking that the file has changed. This can be done by opening ITS.tex with a text editor and saving the document. The manual should then be recompiled using

the "Typeset Preview" command.) From there, the manual may be printed to hardcopy or to a postscript file.

2.2 Document Sections

Each section in the manual corresponds to a separate document in the Docs directory of the repository and may be compiled and viewed individually. Since page and section numbering are performed automatically, these will not be the same as when the entire manual is compiled. The sections (with the respective document name given in parentheses) are:

Introduction to ITS (Introduction.tex) discusses the history of the ITS codes.

Overview of the Documentation (DocsOverview.tex) is this section.

Overview of the ITS Code Package (CodeOverview.tex) gives a brief description of the cross section generators and Monte Carlo codes that comprise the ITS code package and a discussion of the code capabilities available.

Installation (Installation.tex) gives general guidance for installing the ITS software on a platform. It discusses some of the modifications that may be necessary in the configure/make system of files.

Running ITS (RunningITS.tex) contains an overview, as well as step-by-step instructions, for executing a calculation with ITS. Some known platform dependencies of the code are listed.

Code Options (PreprocDefs.tex) contains the preprocessor definitions available for selecting member codes of the Integrated TIGER Series.

Summary of ITS Keywords (Keywordsum.tex) contains tables of input keywords available for multigroup forward, multigroup adjoint, and continuous-energy options and the default settings associated with those keywords.

Keywords for ITS (Keywords.tex) contains an alphabetical listing of the input keywords for all of the ITS codes and descriptions for using each keyword.

TIGER Geometry (GeometryTIGER.tex) contains the formatting requirements for input of 1-D geometry.

CYLTRAN Geometry (GeometryCYLTRAN.tex) contains the formatting requirements for input of 2-D geometry.

ACCEPT Geometry (GeometryACCEPT.tex) contains the formatting requirements for input of 3-D combinatorial geometry.

Output (Output.tex) contains descriptions of the information in each section of an output file.

Suggestions for Efficient Operation (Suggestions.tex) discusses some of the issues involved in using a Monte Carlo code efficiently, such as the proper selection of energy ranges, biasing parameters, and number of particle histories simulated.

P Codes (Pcodes.tex) contains a description of the PCODES code option for detailed ionization and relaxation modeling.

M Codes (Mcodes.tex) contains a description of the MCODES code option for magnetic and electric fields.

Biasing Options and Variance Reduction (Biasing.tex) contains explanations of the biasing settings available in the ITS codes.

Statistics (Statistics.tex) contains a description of how statistical estimations are performed in ITS.

Automatic Subzoning (Subzoning.tex) contains descriptions of how subzoning is implemented in ITS codes.

Random Number Generators (RNGs.tex) contains descriptions of the portable random number generators implemented.

Adjoint Calculations (Adjoint.tex) provides some theoretical descritpion of the adjoint mode of the MITS code option.

Regression Testing of ITS (RegTests.tex) contains step-by-step instructions on how to perform regression tests for ITS and details the tests that comprise the ITS regression test suite.

Summary of ITS-CAD Keywords (CADKeywordsum.tex) contains a table of keywords available for the parameter file used with CAD calculations and default settings associated with those keywords. This is available as a separate document outside of the manual.

Keywords for ITS-CAD (CADKeywords.tex) contains a listing of the parameter keywords for use in the parameter file with the CAD code option and descriptions for using each keyword. This is available as a separate document outside of the manual.

CAD Geometry (GeometryCAD.tex) contains the requirements for input of 3-D CAD geometry models. This is available as a separate document outside of the manual.

Unit Testing of ITS (UnitTests.tex) contains descriptions of each of the unit tests for ITS that are contained in the repository and how to perform those tests. This is available as a separate document outside of the manual.

ACIS Library Build for ITS-CAD (ACISbuild.tex) contains information on building the ACIS libraries and specific modifications that have been necessary to build the libraries on certain platforms. This is available as a separate document and is only available to Sandia developers.

Last Modified: March 18, 2004

3 Overview of the ITS Code Package

ITS consists of three essential code directories:

1. CEPXS – the multigroup cross-section generation program
2. XGEN – the continuous-energy cross-section generation program
3. ITS – the Monte Carlo program

CEPXS is used to generate cross sections for the multigroup ITS codes. A number of physics and modeling decisions can be (or must be) made at the time the multigroup cross sections are generated. XGEN is used to generate cross sections for the continuous-energy ITS codes. Fewer physics and modeling decisions are required when these cross sections are generated, but for example, one must choose between the standard codes and the PCODES. The heart of ITS is the set of Monte Carlo program files. Here, numerous decisions must be made about the simulation, e.g., problem dimensionality, physics options, forward vs. adjoint, biasing options, etc.

3.1 New Capabilities Since 3.0

Here we list the code capabilities available in the ITS codes and equate those to preprocessor definitions and some of the input keyword options. (In the following, ITS refers to the continuous-energy codes, in contrast to MITS, which is the preprocessor directive for the multigroup ITS codes.)

Output options:

KERMA distributions - ITS or MITS forward (PHOTRAN biasing subkeyword)
Dose distributions - ITS or MITS forward
Charge deposition distributions - ITS or MITS forward
Pulse height distributions - ITS (PULSE-HEIGHT keyword)
Point KERMA - MITS (ADJOINT, DETECTOR-RESPONSE/CHARGE keywords)
Point dose - MITS (ADJOINT, DETECTOR-RESPONSE/DOSE keywords)
Point charge - MITS (ADJOINT, DETECTOR-RESPONSE/CHARGE keywords)
Electron escape - ITS or MITS forward or adjoint
Photon escape - ITS or MITS forward or adjoint
Neutron escape - MITS forward or adjoint (User must supply cross sections)
Electron flux - ITS or MITS forward
Photon flux - ITS or MITS forward
Neutron flux - MITS forward (User must supply cross sections)

Code options:

CAD geometry models can be used for three-dimensional transport (**ACCEPT** only) with either ITS photon-only (PHOTRAN biasing) or **MITS** (electron and photon in both forward and adjoint). **CAD** models can be used with all code and output options except **TIGER**, **CYLTRAN**, or ITS electron transport. **CAD** capability currently requires linking with the ACIS geometry libraries.

MPI parallel calculations can be performed with all code and output options except **RNG1**.

MCODES provides the capability of simulating particle transport in electric fields (only in vacuums) and magnetic fields. This capability is only available for ITS with **CYLTRAN** and **ACCEPT**. **MCODES** cannot be used with **MITS** or **PCODES**.

PCODES provides more detailed ionization and relaxation simulation in the continuous-energy codes. For **MITS** a similar capability is the default in the CEPXS generation of the cross sections. The NO-PCODE keyword for CEPXS provides cross sections similar to the ITS standard simulation. To obtain explicit line information with MITS, the LINES keyword for CEPXS must be used.

Last Modified: January 9, 2004

4 Installation

As noted in the Code Overview, there are three components in the ITS code package:

1. CEPXS – the multigroup cross-section generation program
2. XGEN – the continuous-energy cross-section generation program
3. ITS – the Monte Carlo program

We assume that these code directories have been provided. In this section we describe the anticipated *one time only* modifications that may be required to install the ITS package on a new platform. These modifications are made in the configure and Makefile systems. No modifications are anticipated for the codes themselves. Note that the following instructions are written for the ITS code directory, but apply similarly to the XGEN and CEPXS code directories. There should be little difference in the config files for ITS, XGEN, and CEPXS.

WARNING: Before performing any modifications to the files provided, we strongly recommend that an unaltered copy be stored! This will allow for future determinations of code modifications that were necessary.

From within the its/Code directory, execute "./configure". The script will respond with a statement of the form "Configuring for a *-*-* host." Only the last of the three variables (the operating system) is important for configuring. If header and target files exist and are found in the config directory, the configure script will print a statement of the form: Created "Makefile" using "config/mh-*" and "config/mt-*". If the script prints the statement 'Created "Makefile"' but does not state which header and target files were used, then the necessary files do not exist, the configure failed, and the Makefile will not function.

If the configure failed, it will be necessary to create header and target files. The name of the files must be based on the start of the operating system name identified by the configure script.

As an example, the configure script may identify the host as rs6000-ibm-aix4.3.2.0. This indicates that the operating system has been identified as aix4.3.2.0. The configure script will use the config files config/mh-aix and config/mt-aix to create the Makefile. On the other hand, if the configure scripts were to identify the host as i686-unknown-example3.2, then the user would need to create scripts named "config/mh-exa" and "config-mt-exa", or "config/mh-example3.2" and "config/mt-example3.2", or such. It will be easiest to create the new scripts by modifying two of the existing config scripts, so that the required variables and formats are available. The variable ARCH is set in some of the config files for which a timer has been implemented within the ITS codes. If a timer is not available for the platform on which you are installing, ARCH can be left blank, and the code will produce zeroes for timing information.

When new config files are created, it is necessary to modify the configure.in file, so that the configure script can find the new files. The use of an asterisk at the end of the name of the config files will indicate whether the name of the operating system must be identified exactly or if only the given leading characters need to be identified. There are several target files ("config/mt-*") for which the exact name must be identified.

There are times when it is desirable to be able to compile in different ways on a single platform. For example, one may wish at times to compile on a solaris machine to run on that same machine, in which case the host and target machine are the same. Or one may wish to (cross-)compile on the solaris machine to run on another machine, in which case the host and target machines are different. In this case, one can create a configure file for the target machine that can be selected as a flag when configuring. This has been

done for the ASCI Red (tflops) machine. To utilize this, one would execute "configure -target=i386-tflops". If one is adding a new, unique target option, it is necessary to add the option to the configure.in file and to the config.sub file. The changes required in the config.sub file may be identified by searching for "tflops". Specification of a target machine by name can be useful not only for cross compiling but also for installing the software on an unusual instance of an operating system, while maintaining the target config file for the usual operating system.

When the Makefile has been successfully created, one must attempt to make an executable. To do this, it is necessary to specify a valid set of preprocessor definitions. This can be done in either the Makefile or the Makefile.in. Changes made in the Makefile.in will not be lost when one executes configure again. The Makefile.in.defs should not be modified, as it serves as a backup for the Makefile.in. The preprocessor definitions are specified in a section titled "scripted settings". Valid options are listed in this section. Examples of valid options are "RNG = RNG1", "OPT1 = -DMITS", and "OPT2 =".

An executable can then be produced by executing "make". If this is not successful, it will be necessary to identify the settings in the Makefile.in and/or config files that need to be modified. We can offer only our general experience that the problem is (1) most likely to be in the specification of preprocessor definitions in the Makefile.in file or the compilers in the target config file, (2) possibly in other settings in the target config file, and (3) least likely to be in the host config file.

When a working executable is obtained, any modifications required in the Makefile.in file (other than specifying preprocessor definitions) should be transferred to the Makefile.in.defs file, so they will not be lost.

We then recommend that the regression tests be run to determine that the software is functioning properly on the new platform. To run the tests it is necessary to copy the sendn and nmCVS files from the Scripts directory to the SHOME/bin directory. It is also necessary to copy the Subscripts directory to $HOME/bin/Subscripts. If creating a Makefile requires a command other than simply "./configure" without flags, it may be necessary to modify the $HOME/bin/nmCVS script to recognize the platform dependency. A guide for running the regression tests is provided in the section on Regression Testing of ITS.

When the regression tests have been successfully executed, the software may be deemed to have been successfully installed.

Finally, we recommend that the user consider the advantages of importing the software into a CVS version control system[31]. The original files provided should be imported first. Then any modifications required to install the software on the platform can be committed to the repository. This provides a convenient mechanism for storing and tracking future code modifications and extracting changes that have been made to the code over time.

Last Modified: February 2, 2004

5 Running ITS

This document contains outlines for running the ITS code. Instructions are provided for running the code by using commands and running the code by using scripts either with or without access to the CVS repository. Known platform dependencies of the code are discussed in the last section.

5.1 Running ITS without Scripts

CROSS SECTIONS: Cross sections must be generated either with XGEN (for the continuous-energy codes) or with CEPXS (for the multigroup codes).

CHECKOUT: Do a "cvs checkout its" to acquire a copy of the code on on the platform where the repository is located. If necessary, tar the directory and transfer it to the desired platform.

CODE MODIFICATIONS: If desired, code modifications can be made before building an executable.

MAKEFILE SETTINGS: In the directory its/Code, you must alter the Makefile.in settings in the "scripted settings" section to specify the necessary definitions for your build of the ITS executable. There are no valid defaults! An unaltered copy should remain stored as Makefile.in.defs.

CONFIGURE: Execute "./configure". If the code has been previously installed correctly, then the platform and operating system will be identified, and the proper config/mh-* file and config/mt-* file will be included in the Makefile. It may be necessary to specify a target platform when configuring.

MAKE: Execute the Makefile with the "make" command to produce the ITS executable, its.x. If the code has been installed on the platform correctly, the correct commands will be used.

INPUT FILE: An ITS input file must be constructed. Examples are available for each code option in its/Tests/RegTests/Input. See sections on Keywords for ITS for additional information.

CAD FILES: If performing a CAD calculation, a prmfile must be constructed and a satfile must be provided.

EXECUTION: The command "its.x <mdat >output" executes the ITS code, where mdat is the ITS input file. (The files its.x, fort.11, and mdat are required.)
If performing a CAD calculation, the command "its.x prmfile" executes the ITS code. (The files its.x, fort.11, prmfile, mdat, and satfile are required.)

EVALUATE RESULTS: The output file should be evaluated to determine if the run was successful and if the results are satisfactory. For an unsuccessful calculation, an abort call will usually be indicated at the end of the output file. It may be necessary to look for errors in the output file (search for ">>>>>"). The abort may have been postponed until all input had been read, and one error may be the cause of additional errors.

5.2 Running ITS without Scripts - Details

CROSS SECTIONS: Cross sections must be generated either with XGEN (for the continuous-energy codes) or with CEPXS (for the multigroup codes). Unless the FILE-NAMES keyword is used, the file must be named "fort.11". More detailed discussion of generating cross section files can be found with the documentation for the XGEN and CEPXS codes.

CHECKOUT: Do a "cvs checkout -P its" to acquire a copy of the most recent version of the code on the platform where the repository is located. It is recommended that a -P flag be used when checking out a copy of the code to avoid acquiring empty directories that correspond to outdated directory structure. A variety of date and release tags can be used to request older versions of the code. See CVS documentation at http://www.ndim.edrc.cmu.edu/Help/CVS/cvs_toc.html for additional information. If necessary, tar the directory and transfer it to the desired platform.

CODE MODIFICATIONS: If desired, code modifications can be made before building an executable. If you do not have direct access to the CVS repository, it is strongly recommended that an unaltered version of the code be stored for tracking the alterations that you make.

MAKEFILE SETTINGS: In the directory its/Code, you must alter the Makefile.in settings in the "scripted settings" section to specify the necessary definitions for your build of the ITS executable. There are no valid defaults! An unaltered copy should remain stored as Makefile.in.defs. Instructions for setting the preprocessor definitions are included in the "scripted settings" portion of the Makefile.in and in the Code Options section. It is important that options be set under the proper variables (e.g., MITS should be selected under OPT1, MPI should be selected under OPT3, etc.). There are some options that require a -D flag to be included with the definition. These are the variables for which it is valid to leave the definition blank, and they are illustrated in the Makefile.in (e.g., OPT1 can be -DMITS, -DPCODES, or blank).

CONFIGURE: Execute "./configure". If the code has been previously installed correctly, then the platform and operating system will be identified, and the proper config/mh-* and config/mt-* files will be included in the Makefile. It may be necessary to specify a target platform when configuring. See Section 5.5 for information on known platform dependencies.

MAKE: Execute the Makefile with the "make" command to produce the ITS executable, its.x. If the code has been installed on the platform correctly, the correct commands will be used.

INPUT FILE: An ITS input file must be constructed. Examples are available for each code option in its/Tests/RegTests/Input. The keywords relevant to specific code options and their defaults are given in the Summary of ITS Keywords section. Specific instructions for formatting each keyword in the ITS input are given in the Keywords for ITS section.

CAD FILES: If performing a CAD calculation, a prmfile must be constructed and a satfile must be provided. Instructions for setting the CAD parameters in the prmfile file are provided in the Keywords for ITS-CAD section. The satfile should contain the desired CAD geometry in ACIS SAT format.

EXECUTION: The command "its.x <mdat >output" executes the ITS code, where mdat is the ITS input file. (The files its.x, fort.11, and mdat are required to perform a calculation.)

If performing a CAD calculation, the command "its.x prmfile" executes the ITS code. (The files its.x, fort.11, prmfile, mdat, and satfile are required to perform a calculation, with the names of the mdat and satfile specified in the prmfile.)

EVALUATE RESULTS: The output file should be evaluated to determine if the run was successful and if the results are satisfactory. For an unsuccessful calculation, an abort call will usually be indicated at the end of the output file. It may be necessary to look for errors in the output file (search for ">>>>>"). The abort may have been postponed until all input had been read, and one error may be the cause of additional errors. If an error statement is generated, then an anticipated problem has been found in the input and/or cross sections, and diagnostic information should be available. If the code generates an execution error, the user may have introduced a bug via a code modification. If a bug is found in ITS that was not introduced by the user, please notify the ITS developers providing enough details to reproduce and understand the error: a description of the error observed, the version of the code being used, diffs showing any code modifications made, and the input and output files.

5.3 Running ITS with Scripts

CROSS SECTIONS: Cross sections must be generated either with XGEN (for the continuous-energy codes) or with CEPXS (for the multigroup codes).

CHECKOUT: Do a "cvs checkout -P its" to acquire a copy of the code on on the platform where the repository is located. If necessary, tar the directory and transfer it to the desired platform.

BACKUP: Making an unaltered backup copy can be very important for tracking code changes when working on a platform that does not have access to the repository.

SCRIPTS: The scripts sendn and nmCVS that are located in its/Scripts must be copied to $HOME/bin. The scripts in its/Scripts/Subscripts must be copied to $HOME/bin/Subscripts.

CUI FILE: Copy the its/Scripts/its.cui file to your working directory, and edit it for your specific problem. The sections of a cui file are:

1. driver script - the default nmCVS is usually acceptable.
2. defs - selects code options. See Code Options for more information.
3. prmfile - only required for CAD calculations. See Keywords for ITS-CAD for more information.
4. satfile - only required for CAD calculations.
5. diffs - this section must be present. Code modification patches may be specified.
6. mdat - contains the ITS input. See Keywords for ITS for more information.

SENDN: Submit the job using the command "sendn its.cui <jobname>". This script will prompt you for 3 pieces of information (and possibly a 4th depending upon your responses to the first 3). The information consists of the cross section file to be used, whether the code will be compiled in an existing directory or from a cvs checkout, and how to document code modifications.

EXECUTION: If calculations are performed in "interactive" mode (requiring the user to execute the code), the files will be located in $HOME/tmp/<jobname>. The code will be executed as "its.x <mdat >output" for non-CAD calculations and "its.x prmfile" for CAD calculations.

POSTPROC: If calculations are performed in "interactive" mode, the postproc script must be executed in the $HOME/tmp/<jobname> directory to complete the script process.

EVALUATE RESULTS: The output file should be evaluated to determine if the run was successful and if the results are satisfactory. For an unsuccessful calculation, an "ohoh<jobname>.job" file will be returned. Information will be included in the ohoh file that may indicate the source of the error. Additional information may be found in the $HOME/tmp/<jobname> directory. The mlog2 file contains most of the information generated while the scripts were run. If the program compiled successfully but an error was generated during the execution of ITS, an abort call will usually be indicated at the end of the output file. It may be necessary to look for errors in the output file (search for ">>>>>"). The abort may have been postponed until all input had been read, and one error may be the cause of additional errors.

5.4 Running ITS with Scripts - Details

CROSS SECTIONS: The sendn script will request the name of a cross section file. For the continuous-energy codes, the cross section file must be located in $HOME/cross3. For the multigroup codes, the cross section file must be located in $HOME/crossm and must be given a name with the extension ".11". More detailed documentation on generating cross section files can be found with the XGEN and CEPXS codes.

CHECKOUT: Do a "cvs checkout -P its" to acquire a copy of the most recent version of the code on the platform where the repository is located. It is recommended that a -P flag be used when checking out a copy of the code to avoid acquiring empty directories that correspond to outdated directory structure. A variety of date and release tags can be used to request older versions of the code. See CVS documentation at http://www.ndim.edrc.cmu.edu/Help/CVS/cvs_toc.html for additional information.

BACKUP: For working on platforms that do not have direct access to the repository, it is recommended that two copies of the repository be set up: one copy to be used as a working directory in which code modifications and builds can be performed, and one unaltered copy that can be compared with to maintain a record of changes made to the working version. The unaltered copy should be give a name uniquely indicating the cvs version. For example, if the check out was performed as (cvs checkout -D "February 1, 2002" its), then the unaltered copy might be named "its01Feb2002". The name of the directory will be the indication of the version, which is a very important key to repeating a calculation at some later time.

SCRIPTS: The scripts sendn and nmCVS that are located in its/Scripts must be copied to SHOME/bin. The scripts in its/Scripts/Subscripts must be copied to $HOME/bin/Subscripts. Sendn is a script for launching jobs, nmCVS is a driver script for performing jobs, and the Subscripts are utilities used by sendn and nmCVS. Sendn will position the cui file, including the sections of the driver script. The driver script contains the commands necessary to build and execute the program, clean up after itself, and produce relevant result information in a job file. (If necessary, it will also include in the job file information useful in determining the cause of a job failure). $HOME/bin should be included in your $PATH.

CUI FILE: The its/Scripts/its.cui file is available as an example. You can copy it to your working directory, and edit it for your specific problem. The portions of a cui file are:

1. DRIVER SCRIPT: The script nmCVS is available as a driver script. You may substitute a customized script by including it in the first portion of the cui file.

2. DEFS: Instructions for setting the preprocessor definitions are included in the "defs" portion of the its.cui file and in the Code Options section. It is important that options be set under the proper variables (e.g., MITS should be selected under opt1, and MPI should be selected under opt3), because the script will look (i.e., grep) for these specific settings. There should be no spaces in the settings.

   There are several options for running the scripts that may be set in the defs portion of the cui file. Compiler flags may be specified. The user may request email notification that the scripts have finished.

   The user may also request an interactive script. The interactive script should be used on systems that require job queuing or cross compiling. The first half of the script will build the executable. The files for running the job will be located in $HOME/tmp/<jobname>. The executable is named "its.x", the input file is named "mdat", the parameter file for ITS-CAD is named "prmfile", and the program output should be directed to a file named "output". (On janus, the user should move files to scratch space to perform the calculation and move the files back when the calculation is complete.) When the calculation is complete, the user may execute the "postproc" file in the $HOME/tmp/<jobname> directory. This will produce a job file in the directory from which the job was originally submitted.

3. PRMFILE: Instructions for setting the CAD parameters in the "prmfile" portion of the its.cui file are provided in the Keywords for ITS-CAD section. This portion is only required for CAD calculations.

4. SATFILE: The "satfile" portion is used to transfer the desired CAD geometry to the location in the tmp directory where the calculation will be performed. Only two lines are required. The first line should include the desired *.sat file containing the entire CAD geometry. The second line may contain anything but must contain a return. This portion is only required for CAD calculations.

5. DIFFS: Patches to be applied to the code should be inserted in the "diffs" portion of the its.cui file. These patches can be formatted as a cvs diff or as a directory diff. Diffs can be lifted out of job files from previous calculations. Alternatively, one can checkout a copy of the code, make modifications, and then generate a diffs file. On a platform with access to the CVS repository, one can use "cvs diff > diffs" from within the its/Code directory. On a platform without access to the CVS repository one can perform a directory diff, but this must be performed from within the its/Code directory of the unaltered copy of the code, and it must use the -r and -b flags (in that order), such as "diff -r -b . $HOME/itsaltered/Code> diffs". Then, the diffs file can be used in the its.cui file. Multiple diff files (diffs from two different executions of the diff command) cannot be patched to a single cvs file.

   New files can be added to a build. Within the diffs portion, a line of the following format denotes the start of a new file:

   New file: <Directory/Filename>

   To be included in the compiling and linking of the code, the new file must be referenced in the Makefile.in and Makefile.in.defs list of sources. This change in the Makefile.in.defs can also be included in the diffs portion by making the desired change in a copy of Makefile.in.defs and using the cvs diff procedure described above.

6. MDAT: The ITS input should be included in the last portion of the cui file. The keywords relevant to specific code options and their defaults are given in the Summary of ITS Keywords section. Specific instructions for formatting each keyword in the ITS input are given in the Keywords for ITS section.

SENDN WITHOUT CVS: Submit the job using the command "sendn its.cui <jobname>". This script prompts you for the following 3 pieces of information:

1. CROSS SECTIONS: First, it requests the name of the cross section file <cross> to be used. For the continuous-energy codes, the script looks for the file at "$HOME/cross3/<cross>". For the multi-group codes, the script looks for the file at "$HOME/crossm/<cross>.11".

2. LOCAL COMPILE: Second, sendn requests the location of a copy of ITS that can be used for making the executable. Thus, you may use a version of the code that you have checked out and modified. You must give a complete pathname for the base ITS directory (e.g., /scratch/temporary/its). No files will be deleted from the make directory as a result of the calculation, but some files may be modified if requested in the diffs section of the cui file. Any new files that have been included in the directory (other than through the diffs section of the cui file) will not appear in the job file, but they may be used if the Makefile.in.defs has been so modified (in which case, the job file will show the reference to the new file as a difference in the Makefile.in.defs and Makefile.in). Attempts to apply diffs in a directory where the diffs have already been applied for a previous calculations will result in an error.

3. DIRECTORY DIFF: Next, the script requests a local directory with which to perform a diff. The diff command will be used to compare the two directories and all subdirectories. The job file will not record the version number of either directory, therefore the user may not have enough information in the job file to duplicate a calculation unless the diff directory has a name corresponding to the tag used in the cvs checkout of the code. The directory name appears in the job file.

SENDN WITH CVS: Submit the job using the command "sendn its.cui <jobname>". This script prompts you for the following 3 pieces of information (and possibly the 4th depending upon your responses to the first 3):

1. CROSS SECTIONS: First, it requests the name of the cross section file <cross> to be used. For the continuous-energy codes, the script looks for the file at "$HOME/cross3/<cross>". For the multi-group codes, the script looks for the file at "$HOME/crossm/<cross>.11".

2. LOCAL COMPILE: Second, sendn requests the location of a checked-out copy of ITS that can be used for making the executable. Thus, you may use a version of the code that you have checked out and modified. You must give a complete pathname for the base ITS directory (e.g., /scratch/temporary/its). No files will be deleted from the make directory as a result of the calculation, but some files may be modified if requested in the diffs section of the cui file. Any new files that have been included in the directory (other than through the diffs section of the cui file) will not appear in the job file, but they may be used if the Makefile.in.defs has been so modified (in which case, the job file will show the reference to the new file as a difference in the Makefile.in.defs and Makefile.in). Attempts to apply diffs in a directory where the diffs have already been applied for a previous calculations will result in an error.

   CVS COMPILE: To request a cvs checkout of the code, you may respond to this request with "none" (or press "Enter"). The cvs checkout will be performed in the $HOME/tmp/<jobname> directory and should not affect any other versions of the code.

3. CVS DIFF for LOCAL COMPILE: Enter "none" or simply press "Enter" with no input for this third request.

   VERSION for CVS COMPILE: If your response to the second request was "none", the script requests the version for a cvs checkout. The command will be issued as "cvs checkout <options> its/Code". The response to this request will be used for the <options> and any valid cvs flags may be used that do not contain a slash, "/". Examples of valid syntax for responses are: -D now, -D "March 28, 2001", -D "3 hours ago", -D "2 fortnights ago", -r ITSversion5.0, etc. Another valid response is to simply press "Enter" with no input, which will result in the checkout of the most recent version of the code.

4. CVS DIFF for LOCAL COMPILE: If you gave a pathname for making the executable but not for a directory diff, the script requests the version for a cvs diff. The syntax for responses to this request are the same as for specifying the cvs version for a checkout. The directory in which the executable is made will be compared with the repository using the cvs diff command. This version (that will appear in the job file) and the results of the cvs diff (that will also appear in the job file) can be used to reproduce a calculation.

   CVS UPDATE/DIFF for CVS COMPILE: If you did not give a pathname for making the executable, the script requests a version for a cvs update and diff. In this case, the script will check out a version of the code using the options in the third response, apply the "diffs" from the cui file, attempt to update to the version of the code specified in this response, and then compare the resulting code to the repository using the cvs diff command with the options specified in this response. The version requested here (that will appear in the job file) and the results of the cvs diff (that will also appear in the job file) can be used to reproduce a calculation.

   For either of these, if no option is specified for the cvs diff, "-D now" will be used. The date and time of the submission of the calculation, which are recorded in the job file, and the results of the cvs diff can be used to reproduce the calculation at a later date.

EXECUTION: On platforms where the interactive script is used because cross compiling is required, it may be necessary to move the files to perform the calculation. However, it will be necessary to move the files back to their original location after execution. For example, calculations on janus are much more efficient if performed in /scratch space, but the postproc script will not work on janus. After execution, the files must be moved back to $HOME/tmp space, and the postproc script must be executed on sasn100.

If execution fails or if after the run the output file is found to contain errors, it may be desirable to use the tmp directory as a working directory to debug the code or calculation. However, after the calculation has been performed successfully, it is likely that one must resubmit the job with corrected input so that the job file will accurately reflect the code modifications and the input used to perform the calculation.

POSTPROC: When the interactive script is used, postproc must be executed to produce a job file. The job file will be placed in the directory from which the job was originally launched, just as with the non-interactive script.

EVALUATE RESULTS: Errors may occur at a number of stages in the calculation. The stage at which the error occurred is usually indicated near the start of the ohoh file. Some common causes of errors are:

1. Configure - If the platform has not been used before, the necessary config files may not be present.

2. Make - If the defs have not been properly specified or Makefile.in.defs has been modified, the definitions may not have been properly set by the scripts. The "scripted settings" section of the Makefile.in.defs must contain certain words to be replaced by the scripts.

3. Execution - If an error statement is generated (search in the output for ">>>>>"), then an anticipated problem has been found in the input and/or cross sections, and diagnostic information should be available. Such an error might also result if the Makefile.in.defs was modified, and an executable with the wrong code options is being used. If the code generates an execution error, the user may have introduced a bug via a code modification. If a bug is found in ITS that was not introduced by the user, please notify the ITS developers providing enough details to reproduce and understand the error: a description of the error observed, the version of the code being used, diffs showing any code modifications made, and the input and output files.

5.5 Platform Dependencies

The following is a list of platforms on which ITS has been successfully built and tested. Following the name of the platform is the system type and operating system.

- Scorpio and Virgo (IBM RS6000, AIX4)

  The driver script is functional for serial versions of the code. ACIS6 libraries are available for CAD. The AIX architecture option supplies a cpu timer.

- Blue Pacific and Frost (IBM, AIX5.1)

  The driver script is functional for building the executable on a compile node in interactive mode.

  Jobs are submitted to the queue using the psub command:

  psub -ln <nodes> -g <processors> -tM <max time> -noDFS its.com

  If not using the scripts, the configure command must specify Blue Pacific as the target platform:

  configure -target=powerpc-bluepacific

  ACIS6 is available for CAD calculations. Executables built with compiler optimization will not function. For the code to function properly, the debug flag must be used for compiling the ITS C++ code, and this is set in the bluepac configure file.

- White (IBM, AIX5.1)

  The driver script is functional for building the executable on a compile node in interactive mode.

  Jobs are submitted to the queue using the psub command:

  psub -ln <nodes> -g <processors> -tM <max time> -noDFS its.com

  If not using the scripts, the configure command must specify White as the target platform:

  configure -target=rs6000-white

  ACIS6 is available for CAD calculations. Executables built with compiler optimization will not function. For the code to function properly, the debug flag must be used for compiling the ITS C++ code, and this is set in the white configure file.

- Taos (Sun, Solaris 5.7)

  The driver script is functional for non-CVS commands. With MPI, the script is set to use 2 processors. The mpirun command should be used to specify the number of processors for parallel calculations. CAD options are not available. The SUN architecture option supplies a cpu timer in serial.

- Luigi (Sun, Solaris 5.8)

  The driver script is functional for non-CVS commands. The scripts can be used in either serial or MPI. The scripts can be used in either interactive or non-interactive mode. With MPI, the script is set to use 2 processors.

  A queuing system is available, and MPI jobs can be submitted as:

  bsub -n<# processors> -i<input file> -o<output file> mpijob mpirun its.x

  CAD options are not available. The SUN architecture option supplies a cpu timer in serial.

- Janus (i386, TFLOPS), compile on sasn100 (Sun, Solaris 5.9)

  The driver script is functional for building the executable on sasn100 in interactive mode. The job must be executed from janus. The post-processing must be done on the application server.

  Jobs are run interactively using the yod command:

  yod -sz <# processors> -p 3 its.x < <input file> > <output file> &

  Jobs may be submitted to the queuing system with the qsub command.

  If not using the scripts, the configure command must specify janus as the target platform:

  configure -target=i386-tflops

  ACIS6 libraries are available for CAD. For CAD to be functional, the ASCI_RED architecture must be specified.

- Janus-s (i386, TFLOPS), compile on sasn101 (Sun, Solaris 5.9)

  The code functions as on Janus with compiling on sasn101. The configure command is:

  configure -target=i386-tflopss

- QA, QB (HP Alpha, Tru64 OSF1 V5)

  The driver script is functional for building the executable in interactive mode. ACIS6 libraries are available for CAD. Jobs are submitted to the queuing system using the bsub command. The ACIS library path must be set in the config/mt-osf5 file for running on QSC or QT.

- Zelda (Compaq Alpha, Tru64 OSF1 V5)

  The driver script is functional for non-CVS commands, but should only be used in interactive mode with jobs submitted to the queuing system. The scripts will function much more efficiently if the $HOME variable is set to your /scratch space since communications with /remote space are very slow.

  Stdin cannot be read on Zelda, so it is recommended that the ITSINP definition be used. This will open the input file as "its.inp" and the output file as "its.out". The user may wish to alter the code to use the names "mdat" and "output" to conform with naming conventions in the scripts.

  Jobs are submitted to the queuing system as:

  bsubrms -n<# processors> -o<output file> its.x

  If not using the scripts, the configure command must specify zelda as the target platform:

  configure -target=alpha-zelda

  CAD options are not available.

- Ross (Compaq Alpha, Tru64 OSF1 V5), compile on Angara

  The driver script is functional for building the executable in interactive mode. The job must be executed from ross. ACIS6 libraries are available for CAD. ITS-CAD must be executed from ross2 if the executable is larger than 16MB. The post-processing must be done on angara.

  Jobs are run interactively using the yod command:

  yod -sz <# processors> its.x < <input file> > <output file> &

  Jobs may be submitted to the queuing system with the qsub command.

  If not using the scripts, the configure command must specify ross as the target platform:

  configure -target=alpha-ross

- Gollum (DEC Alpha, OSF1 V4)

  The driver script is functional for serial versions of the code. ACIS6 libraries are available for CAD. The OSF1 architecture option provides a wall-clock timer.

- Linux (x86, Redhat Linux 7.X)

  The driver script is functional.

  Severe run-time problems have been observed using optimization with gnu compilers prior to gcc version 3.0.4. Erroneous results have been observed using gnu compilers after gcc version 3.0.4.

  CAD options are not generally available.

- Crater (AMD Athlon, Linux)

  The driver script is functional in all modes. With MPI, the script is set to use 2 processors. The mpirun command should be used to specify the number of processors for parallel calculations. Computationally intense calculations should be run on compute nodes. Either the calculations should be submitted while logged into a compute node, or the executable should be compiled with the DYNAMIC option so that the master process does not perform Monte Carlo simulations. To perform calculations on the compute nodes, the files must be located within the /chome/bin directory, which is mounted across all compute nodes.

  Severe run-time problems have been observed using optimization with gnu compilers prior to gcc version 3.0.4. Erroneous results have been observed using gnu compilers after gcc version 3.0.4.

- PC

  No configuration, Makefile, regression tests, or other scripts are available.

  There has been limited testing on this platform.

Last Modified: January 9, 2004

6 Code Options

Numerous code options can be selected for compiling the ITS codes. These options have been implemented as preprocessor definitions. It is necessary to choose between the multigroup (MITS) codes and the continuous-energy codes. (The selection between forward and adjoint mode in the MITS codes is made as an input option.) Other options include choosing between the 1-, 2-, and 3-dimensional geometry representations and choosing a random number generator. The platform setting is specified in the config/mt-* file, so the user need not be concerned about these definitions unless creating configure files for a new platform.

The code options must be specified in either the CUI file (if using the scripts to execute the code) or in the "scripted settings" section of the Makefile (if building an executable). Refer to the section on Running ITS for more details on applying definitions.

6.1 Preprocessor Definitions

MITS (to request the multigroup codes; omitted to request the continuous energy codes)

The platform may be selected as one of the following (though the setting is generally included in the config/mt-* file, so this is automatically selected by executing configure):
AIX (provides a CPU-timer and error traceback)
PC (provides a CPU-timer)
SUN (provides a CPU-timer)
OSF1 (provides a wall-clock timer)
LINUX (provides a wall-clock timer)
ASCI_RED (unique implied do-loop structure required for CAD on ASCI_RED)
(MPI provides a wall-clock timer independent of the platform.)

Select the code as one of:
TIGER (1-D)
CYLTRAN (2-D cylindrical geometry; 3-D transport)
ACCEPT (3-D)

Select the random number generator as one of:
RNG1 (generator in 3.0, only available in serial)
RNG2 (RANMAR)
RNG3 (Mersenne Twister, only available in development version)

To select to run on a parallel platform:
MPI
To select dynamic parallel load balancing:
DYNAMIC

Other available options are:
PCODES (more ionization and relaxation for continuous energy codes)
MCODES (magnetic and electric fields for continuous energy codes)
CAD (hooks must be provided to a CAD package)
ITSINP (the input file is opened as "its.inp", the output file is opened as "its.out")

6.2 Definition Requirements

TIGER, **CYLTRAN**, or **ACCEPT** must be selected as the code.
RNG1, **RNG2**, or **RNG3** must be selected as the random number generator.
CYLTRAN is not currently functional for **MITS**.
PCODES or **MCODES** cannot be used with **MITS**.
PCODES and **MCODES** are mutually exclusive.
MCODES cannot be used with **TIGER**.
RNG1 cannot be used with **MPI**.
CAD can only be used with **ACCEPT**.
If none of **AIX**, **PC**, **SUN**, **OSF1**, **LINUX**, and **MPI** are selected, there will be no timer.

Last Modified: March 3, 2004

7 Summary of ITS Keywords

This section contains listings of keywords relevant to the ITS (continuous-energy) codes, the MITS codes in forward mode, and the MITS codes in adjoint mode in Tables 3, 4, and 5, respectively. Only those keywords applicable to the code and mode are listed in each table. Many keywords in forward mode do not apply to adjoint mode and vice versa. The keywords are listed approximately in order of importance. In addition, for each keyword the default code behavior is listed. The default behavior will be employed by the code if the keyword is not found in the input deck.

The secondary keywords used for biasing are listed in Table 6. These are secondary keywords for the BIASING, BIAS-GLOBAL, and BIAS-ZONE keywords. The subkeyword, limitations on the code and mode with which the subkeyword can be used, and the default code behavior are listed in the table. In addition, the global and zone-dependent features of the subkeyword are listed. The BIAS-GLOBAL and BIAS-ZONE keywords are intended to be used together. The BIAS-GLOBAL subkeywords are used to set global biasing parameters. The BIAS-ZONE subkeywords are used to set zone-dependent biasing parameters or to activate global parameters on a zone-dependent basis. The BIASING keyword is an alternative to the BIAS-GLOBAL and BIAS-ZONE functionality and allows the user to set global and zone-dependent parameters within the same subkeywords.

More detailed descriptions of the syntax, subkeywords, and use of these keywords are contained in the Keywords for ITS section. In some cases, these keywords or their defaults depend upon the code option (preprocessor definition) beyond the choice of MITS or ITS. A listing of the available preprocessor definitions is contained in the section on Code Options.

Table 3. ITS keywords and default settings

KEYWORD	DEFAULT
**** GEOMETRY ****	
GEOMETRY	required
ESCAPE-SURFACES	all escape surfaces
**** SOURCE ****	
ELECTRONS or PHOTONS	electron source
ENERGY or SPECTRUM	1.0 MeV monoenergetic
POSITION	point source at origin (TIGER and ACCEPT) on axis at minimum-z (CYLTRAN)
DIRECTION	monodirectional source in positive-z direction
CUTOFFS	electrons: 5% of maximum; photons: 0.01 MeV
**** OUTPUT OPTIONS ****	
ELECTRON-EMISSION	off
ELECTRON-ESCAPE, PHOTON-ESCAPE	off
ELECTRON-FLUX, PHOTON-FLUX	off
PULSE-HEIGHT	off
**** COMMONLY USED OPTIONS ****	
ECHO	off
TITLE	blank title
HISTORIES or HISTORIES-PER-BATCH	1000 histories
BATCHES	20 batches
TASKS	number of processors available (MPI Only)
BIASING or BIAS-GLOBAL and BIAS-ZONE	no biasing parameters are activated
**** RARELY USED OPTIONS ****	
DUMP and/or RESTART	dump off; no restart
PRINT-ALL or NO-INTERMEDIATE-OUTPUT or NO-DEPOSITION-OUTPUT or NO-SZDEPOSITION-OUTPUT	final batch in output; intermediate in fort.12
RANDOM-NUMBER	0 (converted to 5^{19})
NEW-DATA-SET	1 run
FILE-NAMES	default names (fort.3, fort.11, etc.)
FINITE-ELEMENT-FORMAT	no fort.3 output (ACCEPT only)
REFLECTION-ZONE	no reflection zone (ACCEPT only)
DEPOSITION-UNITS	dose in MeV per source particle (ACCEPT only) charge deposition in electrons per source particle
CUTOFF-PHOTONS-ESCAPE	energy of cutoff photons is deposited locally
**** DEVELOPMENT USE ONLY ****	
DETAIL-IONIZE	source of ionization is not detailed
NO-COHERENT	photon coherent scattering is simulated
NO-INCOH-BINDING	binding effects in incoherent scattering are included
NO-KICKING	terminal processing includes kicking
NO-KNOCKONS	secondary knock-on electrons
NO-STRAGGLING	energy-loss straggling
RESTART-HISTORY	no restart
SIMPLE-BREMS	more accurate bremsstrahlung distributions

Table 4. MITS forward keywords and default settings

KEYWORD	DEFAULT
**** GEOMETRY ****	
GEOMETRY	required
ESCAPE-SURFACES	all escape surfaces
**** SOURCE ****	
ELECTRONS or PHOTONS or NEUTRONS	electron source
ENERGY or SPECTRUM	mono-group source in the highest-energy group
POSITION	point source at origin
DIRECTION	monodirectional source in positive-z direction
CUTOFFS	bottom of the lowest-energy group for each species
**** OUTPUT OPTIONS ****	
ELECTRON-ESCAPE	off
ELECTRON-FLUX	off
PHOTON-ESCAPE	off
PHOTON-FLUX	off
NEUTRON-ESCAPE	off
NEUTRON-FLUX	off
**** COMMONLY USED OPTIONS ****	
ECHO	off
TITLE	blank title
HISTORIES or HISTORIES-PER-BATCH	1000 histories
BATCHES	20 batches
TASKS	number of processors available (MPI only)
BIASING or BIAS-GLOBAL and BIAS-ZONE	no biasing parameters are activated
**** RARELY USED OPTIONS ****	
DUMP and/or RESTART	dump off; no restart
PRINT-ALL or NO-INTERMEDIATE-OUTPUT or NO-DEPOSITION-OUTPUT or NO-SZDEPOSITION-OUTPUT	final batch in output; intermediate in fort.12
RANDOM-NUMBER	0 (converted to 5^{19})
NEW-DATA-SET	1 run
MICRO	deposition calculated via flux-folding
FILE-NAMES	default names (fort.3, fort.11, etc.)
FINITE-ELEMENT-FORMAT	no fort.3 output (ACCEPT only)
REFLECTION-ZONE	no reflection zone (ACCEPT only)
DEPOSITION-UNITS	dose in MeV per source particle (ACCEPT only) charge deposition in electrons per source particle
CUTOFF-PHOTONS-ESCAPE	energy of cutoff photons is deposited locally
**** DEVELOPMENT USE ONLY ****	
RESTART-HISTORY	no restart

Table 5. MITS adjoint keywords and default settings

KEYWORD	DEFAULT
ADJOINT	forward
**** GEOMETRY ****	
GEOMETRY	required
**** DETECTOR ****	
DETECTOR-RESPONSE	required
**** SOURCE OUTPUT OPTIONS ****	
SOURCE-SURFACES	all escape surfaces
SPECTRUM	only a flat forward spectrum is used
ELECTRON-SURFACE-SOURCE	off
ELECTRON-VOLUME-SOURCE	off (not functional)
PHOTON-SURFACE-SOURCE	off
PHOTON-VOLUME-SOURCE	off (not functional)
NEUTRON-SURFACE-SOURCE	off
NEUTRON-VOLUME-SOURCE	off (not functional)
**** COMMONLY USED OPTIONS ****	
ECHO	off
TITLE	blank title
HISTORIES or HISTORIES-PER-BATCH	1000 histories
BATCHES	20 batches
TASKS	number of processors available (MPI only)
BIASING or BIAS-GLOBAL and BIAS-ZONE	no biasing parameters are activated
**** RARELY USED OPTIONS ****	
CUTOFFS	top of the highest-energy group for each species
DUMP and/or RESTART	dump off; no restart
PRINT-ALL or NO-INTERMEDIATE-OUTPUT	final batch in output; intermediate in fort.12
RANDOM-NUMBER	0 (converted to 5^{19})
NEW-DATA-SET	1 run
FILE-NAMES	default names (fort.11, fort.12, etc.)
REFLECTION-ZONE	no reflection zone (ACCEPT only)
**** DEVELOPMENT USE ONLY ****	
RESTART-HISTORY	no restart

Table 6. Biasing subkeywords, default settings, and properties

KEYWORD	DEFAULT	GLOBAL/ZONE PROPERTIES
COLLISION-FORCING	natural photon interactions	no global settings; specified by zone
ELECTRON-RR (Forward Only)	natural number of photon-produced secondary electrons are followed	global Russian Roulette probability; activated by zone
NEXT-EVENT-ESCAPE	off, unless using PHOTON-ESCAPE or PHOTON-SURFACE-SOURCE	global
PHOTRAN (Forward Only)	secondary electrons are tracked	no secondary electrons set globally; exceptions by zone
SCALE-BREMS (ITS Only)	natural bremsstrahlung production	global scaling factor; activated by zone (activates SCALE-IMPACT)
SCALE-EP (MITS Only)	natural electron-to-photon production	global scaling factor; activated by zone
SCALE-IMPACT (ITS Only)	20% of brems scaling if used; otherwise, natural impact ionization	global scaling factor; activated in zones with SCALE-BREMS
SCALE-PE (MITS Only)	natural photon-to-electron production	global scaling factor; activated by zone
TRAP-ELECTRONS (Forward Only)	no trapping above cutoff energy	both global and local; the more stringent applies
**** DEVELOPMENT USE ONLY ****		
ELECTRAN (Forward Only)	secondary and scattered photons are tracked	no secondary/scattered photon set globally; exceptions by zone
NO-BANK (MITS Only)	secondary particles are banked	Global

Last Modified: March 3, 2004

8 Keywords for ITS

The input keywords must be specified in either the CUI file (if using the scripts to execute the code) or in the input file (if executing the code manually). Refer to the documentation on Running ITS for more details on specifying the input file in the CUI file or otherwise.

8.1 Input Notation

The keywords that are appropriate to use depend upon the code options that have been selected in building the executable (and whether the MITS code is being run in forward or adjoint mode). An overview of the preprocessor definitions is available in the Code Options section. An overview of the keywords that apply to MITS forward, MITS adjoint, and the forward continuous-energy ITS codes is available in the Summary of ITS Keywords section. In this section, following each keyword are any restrictions on the code options or mode. The designation "ITS Only" refers to the continuous-energy codes (i.e., not MITS). The designation "Forward Only" refers to both the MITS and ITS codes, unless otherwise noted. All other designations refer to the preprocessor definitions used in building the executable.

Most primary keywords are order-independent. The two exceptions to this are the NEW-DATA-SET and SUBZONE-ONLY usage of the GEOMETRY keyword.

Most keywords must be used once and not repeated in an input file. Exceptions are BIAS-ZONE, ECHO, and NEW-DATA-SET. Most sub-keywords should be used only once per use of their primary keyword. Exceptions are the sub-keywords of ESCAPE-SURFACES, SOURCE-SURFACES, and GEOMETRY.

Parameters are associated with the preceding keyword appearing on the same line. If parameters are omitted, they will be set to zero. Consideration should be given to the fact that in some situations this value is invalid and will trigger an error. Values expected on lines following a keyword are not optional, unless otherwise stated.

Comments may be inserted in the input deck. Anything appearing to the right of an asterisk anywhere in the input deck will be treated as a comment and ignored by the code.

Input is not case sensitive, with only one significant exception. File names entered with the FILE-NAMES keyword will be used exactly as provided in the input deck. Character input provided with the TITLE and DEPOSITION-UNITS keywords will appear in the output file exactly as provided, but the case will not affect the performance of the code.

8.2 Keywords

1. **ADJOINT** *(MITS Only)*

   Syntax: ADJOINT

   Example: ADJOINT

   Default: Forward

   This keyword triggers adjoint transport. It can be used at any point in the input deck. Forward and adjoint runs can be mixed, but for adjoint calculations the adjoint specification must be made for each new-data-set.

   Adjoint mode requires 3 mandatory keywords: GEOMETRY, DETECTOR-RESPONSE, and a surface or volume source. The quantity of interest must be specified using the DETECTOR-RESPONSE

keyword with one of the secondary keywords CHARGE, DOSE, ESCAPE or KERMA. The forward source(s) must be described using one or more of the source keywords with prefix PHOTON-, ELECTRON-, or NEUTRON- and suffix SURFACE-SOURCE or VOLUME-SOURCE. (Volume sources are not yet implemented.) If a surface-source is requested, the user may further select surfaces using the SOURCE-SURFACES primary keyword. This must be done for ACCEPT (no defaults); for TIGER, the default is that both surfaces are specified.

2. **BATCHES**

   Syntax: BATCHES [parameter(1)]

   Example: BATCHES 10

   Default: 20 batches

Number of batches of primary particles to be run. [parameter(1)] batches are performed in order to obtain estimates of statistical uncertainties. Each batch contains an equal number of source particles selected either by the histories keyword or by the histories-per-batch keyword. Accuracy of the estimates degrades substantially for fewer than 10 batches. Although increasing the number of batches improves this accuracy (for a given number of histories per batch), it also increases the overhead (run time).

3. **BIASING**

   Syntax: BIASING

   Example: BIASING

   Default: No biasing parameters are activated. Electron trapping is determined by the global parameter. Secondary electrons are followed globally (if electrons are included in the cross sections).

Selectively turns on the input-zone-dependent bias parameters specified (Photon Collision Forcing, Photran, Russian Roulette, Scale-Electron-to-Photon Interactions, Scale-Photon-to-Electron Interactions, and/or Electron Trapping). This primary keyword may be repeated, but some secondary keywords should not be repeated.

Restrictions: ELECTRON-RR is disallowed in adjoint. The user cannot simultaneously specify SCALE-EP and SCALE-PE in the same input zone. Note that SCALE-EP and SCALE-PE always refer to scaling the forward cross section, even in adjoint. Thus, when using SCALE-PE in adjoint, an electron-adjuncton would be more likely to turn into a photon-adjuncton.

The keyword BIASING enables the same global biasing functions as BIAS-GLOBAL and the same local biasing functions as BIAS-ZONE. These are alternative methods for setting biasing parameters. Neither BIAS-GLOBAL nor BIAS-ZONE can be used with BIASING.

(a) **COLLISION-FORCING**

Syntax: COLLISION-FORCING [parameter(1)]

Example: COLLISION-FORCING 5

1 4-6 8

0.3 0.2 0.1 0.5 0.1

Default: Photon cross sections determine interaction probabilities.

This keyword specifies the photon forced interaction probabilities. This keyword must be followed by two sets of [parameter(1)] numbers. The first set contains the input zones for which forced interaction probabilities are to be specified. The second set specifies the forced interaction probabilities for the corresponding input zones.

(b) **ELECTRAN** *(Forward Only)*

Syntax: ELECTRAN

Example: ELECTRAN

1, 4-6, 8-

Default: Secondary and scattered photons are followed. If photon cross sections are not provided for MITS, then this keyword is unnecessary.

This keyword indicates that electron-produced secondary photons are not to be tracked and scattered photons are not to be tracked. (Coherent scattering of photons is allowed; the NO-COHERENT keyword can be used to deactivate this physics.)

Additional parameters are optional and specify exceptions. That is, the zones in which secondary photons are to be tracked are listed beginning on the following line. A dash indicates that all zones between two numbers are included. Beginning the list with a dash includes all zones from 1 to the indicated zone. Ending the list with a dash includes all zones from the last number given to the last zone number.

The ELECTRAN secondary keyword should not be repeated. Using ELECTRAN with a photon source provides the equivalent of a first-collision electron source. Using ELECTRAN with an electron source provides electron-only transport.

(c) **ELECTRON-RR** *(Forward Only)*

Syntax: ELECTRON-RR [parameter(1)] [keyword]

Example: ELECTRON-RR 0.1 CUSTOM-RR

1 4-6 8 9

Default: Russian Roulette is not used. Natural photon-to-electron cross sections are used.

[parameter(1)] is the Russian Roulette survival probability used in determining the number of photon produced secondary electrons followed. If [parameter(1)] is omitted or 0.0, Russian Roulette will be used such that the natural number of electrons (the number produced if SCALE-BREMS or SCALE-EP had not been used) would be followed, if Russian Roulette and SCALE-BREMS or SCALE-EP were used throughout the problem.

The keyword **CUSTOM-RR** indicates that customized Russian Roulette logic has been included by the user in function FLRRK.

The additional parameters set the zones for which Russian Roulette is to be turned on. The list of input zones beginning on the following line specify the regions of the problem where Russian Roulette is to be used. The fraction of photon produced secondary electrons to be followed is the inverse of the scaled bremsstrahlung production.

(d) **NEXT-EVENT-ESCAPE**

Syntax: NEXT-EVENT-ESCAPE

Example: NEXT-EVENT-ESCAPE

Default: Feature is off, unless photon-escape or photon-surface-source is specified.

With this keyword, a more efficient calculation of integral photon escape can be made. For differential escape scroring, this feature is automatic, and this keyword is redundant.

(e) **NO-BANK** *(MITS Only)*

Syntax: NO-BANK

Example: NO-BANK

Default: Secondary particles are banked and relative weights are kept at unity.

With this keyword, secondary particles are not banked. Rather, their weights are changed to account for multiplicity and absorption.

(f) **PHOTRAN** *(Forward Only)*

Syntax: PHOTRAN

Example: PHOTRAN

1, 4-6, 8-

Default: Secondary electrons are followed. If electron cross sections are not provided for MITS, then this keyword is unnecessary.

The keyword PHOTRAN indicates that photon-produced secondary electrons are not to be tracked. Additional parameters are optional and specify exceptions. The zones in which secondary electrons are to be tracked are listed beginning on the following line. A dash indicates that all zones between two numbers are included. Beginning the list with a dash includes all zones from 1 to the indicated zone. Ending the list with a dash includes all zones from the last number given to the last zone number. The PHOTRAN secondary keyword should not be repeated. Note: An electron source can be used with PHOTRAN since it only excludes tracking of secondary electrons.

(g) **SCALE-BREMS** *(ITS Only)*

Syntax: SCALE-BREMS [parameter(1)] [parameter(2)]

Example: SCALE-BREMS 500.0 2

4-6, 9

Default: Natural bremsstrahlung cross sections are used.

[parameter(1)] is a scale factor used to modify bremsstrahlung production so as to increase the photon population without increasing the number of primary histories. For example, if [parameter(1)] is set equal to two, then there will be twice as much bremsstahlung photon production. The ELECTRON-RR secondary keyword can be used to control the number of secondary electrons generated by this increased population of photons.

[parameter(2)] is the index of the material, according to the order in which the materials are read from the cross section file, on which the impact ionization scaling is based if SCALE-IMPACT is not used. The default is material number 1.

The additional parameters beginning on the following line specify those zones in which bremsstrahlung biasing is activated.

(h) **SCALE-EP** *(MITS Only)*

Syntax: SCALE-EP [parameter(1)]

Example: SCALE-EP 2.0

4, 5, 6, 8-9

Default: Natural electron-to-photon cross sections are used.

This keyword specifies the factor by which the electron-to-photon cross sections are to be scaled and in which input zones the scaled cross sections are to be used. [parameter(1)] is the scaling

factor. The additional parameters beginning on the following line are the input zones in which the scaled cross sections are applied. Because the scaling factor must only be set once, this secondary keyword may not be repeated.

(i) **SCALE-IMPACT***(ITS Only)*

Syntax: SCALE-IMPACT [parameter(1)]

Example: SCALE-IMPACT 20.0

Default: Natural probability of electron impact ionization, except that if the SCALE-BREMS keyword has been used, the scale factors for electron impact ionization will be based on the bremsstrahlung scaling (such that for an electron slowing from the maximum energy to the global electron cutoff energy for every five bremsstrahlung interactions there will be one electron impact ionization event in the material specified by [parameter(2)] of SCALE-BREMS).

[parameter(1)] is used to scale electron impact ionization so as to increase the photon population (line radiation) without increasing the number of primary histories. The cross sections are scaled such that an electron slowing from the maximum energy to the global electron cutoff energy will, on the average, undergo a number of ionization events equal to [parameter(1)] in each material. The values by which the cross sections may be scaled in every material are written to output. The cross sections will never be scaled down (such that fewer ionization events are simulated than with the natural cross sections).

WARNING: Impact ionization scaling is only activated in zones in which SCALE-BREMS is activated.

(j) **SCALE-PE** *(MITS Only)*

Syntax: SCALE-PE [parameter(1)]

Example: SCALE-PE 0.5

3 7

Default: Natural photon-to-electron cross sections are used.

This keyword specifies the factor by which the photon-to-electron cross sections are to be scaled and in which input zones the scaled cross sections are to be used. [parameter(1)] is the scaling factor. The additional parameters are the input zones in which the scaled cross sections are applied. Because the scaling factor must only be set once, this secondary keyword may not be repeated.

(k) **TRAP-ELECTRONS** *(Forward Only)*

Syntax: TRAP-ELECTRONS [parameter(1)] [parameter(2)] [keyword]

Example: TRAP-ELECTRONS 42 5 CUSTOM-TRAP

1 4-6 8

40 35 40 35 35

Default: The trap-electron energy group is the electron cutoff group.

For ITS, the global electron trapping cutoff specified by [parameter(1)] is energy in MeV.

For MITS, the global electron trapping cutoff is the lower energy bound of the group specified by [parameter(1)].

This keyword must be followed by two sets of [parameter(2)] numbers. The first set contains the input zones for which electron trapping energies are to be specified. The second set specifies the energy below which trapping is tested for the corresponding input zones. In ITS the electron

trapping energy is specified directly in MeV. In MITS the corresponding energy group is specified. (The lower energy group bound is used in forward, and the upper energy group bound is used in adjoint.)

Electron trapping is either ineffective or not implemented correctly yet for adjoint calculations.

Electron trapping is not fully functional for CAD. Trapping will only function with "GEOMETRY 3" and "GEOMETRY 4". Trapping will be performed only on subzones that do not include a zone boundary.

The keyword **CUSTOM-TRAP** indicates that customized trapping logic has been included by the user in subroutine SAVE.

4. **BIAS-GLOBAL**

Syntax: BIAS-GLOBAL

Example: BIAS-GLOBAL

Default: No biasing parameters are activated. Secondary electrons are followed globally (if electrons are included in the cross sections).

The sub-keywords ELECTRON-RR, SCALE-BREMS, SCALE-EP, and SCALE-PE may be used to set global biasing parameters, but these parameters will only be used if activated on a zone-by-zone basis with the BIAS-ZONE keyword. The BIAS-ZONE keyword can also be used to specify exceptions to the global PHOTRAN setting. Zone-dependent trapping energies that are more stringent than the global setting may be specified with the BIAS-ZONE keyword.

The keywords BIAS-GLOBAL and BIASING are alternative methods for setting global biasing parameters and may not both be used.

(a) **ELECTRAN** *(Forward Only)*

Syntax: ELECTRAN

Example: ELECTRAN

Default: All secondary photons are followed. If photon cross sections are not provided for MITS, then this keyword is unnecessary.

The keyword ELECTRAN indicates that no electron-produced secondary photons are to be followed and no scattered photons are to be followed. (Coherent scattering of photons is allowed; the NO-COHERENT keyword can be used to deactivate this physics.) If no exception zones are specified with the BIAS-ZONE/ESEC keyword, no electron-produced secondary photons are tracked in any part of the problem.Using ELECTRAN with a photon source provides the equivalent of a first-collision electron source. Using ELECTRAN with an electron source provides electron-only transport.

(b) **ELECTRON-RR** *(Forward Only)*

Syntax: ELECTRON-RR [parameter(1)] [keyword]

Example: ELECTRON-RR 0.1 CUSTOM-RR

Default: The natural number of photon produced secondary electrons will be followed.

[parameter(1)] is the Russian Roulette survival probability used in determining the number of photon produced secondary electrons to be followed. If [parameter(1)] is omitted or 0.0, Russian Roulette will be used such that the natural number of electrons (the number produced if SCALE-BREMS or SCALE-EP had not been used) would be followed, if Russian Roulette and SCALE-BREMS or SCALE-EP were used throughout the problem.

The input zones in which Russian Roulette is applied must be specified with the BIAS-ZONE keyword. Because the scaling factor must only be set once, this secondary keyword may not be repeated.

The keyword **CUSTOM-RR** indicates that customized Russian Roulette logic has been included by the user in function FLRRK.

(c) **NEXT-EVENT-ESCAPE**

Syntax: NEXT-EVENT-ESCAPE

Example: NEXT-EVENT-ESCAPE

Default: Feature is off, unless photon-escape or photon-surface-source is specified.

With this keyword, a more efficient calculation of integral photon escape can be made. For differential escape scroring, this feature is automatic, and this keyword is redundant.

(d) **NO-BANK** *(MITS Only)*

Syntax: NO-BANK

Example: NO-BANK

Default: Secondary particles are banked and relative weights are kept at unity.

With this keyword, secondary particles are not banked. Rather, their weights are changed to account for multiplicity and absorption.

(e) **PHOTRAN** *(Forward Only)*

Syntax: PHOTRAN

Example: PHOTRAN

Default: All secondary electrons are followed. If electron cross sections are not provided for MITS, then this keyword is unnecessary.

The keyword PHOTRAN indicates that no photon-produced secondary electrons are to be followed. If no exception zones are specified with the BIAS-ZONE/PSEC keyword, no photon-produced secondary electrons are tracked in any part of the problem. The PHOTRAN keyword should not be repeated. An electron source can be used with PHOTRAN since it only excludes tracking of photon-produced secondary electrons.

(f) **SCALE-BREMS** *(ITS Only)*

Syntax: SCALE-BREMS [parameter(1)] [parameter(2)]

Example ITS: SCALE-BREMS 500. 2

Default: Bremsstrahlung production (and photon-generated electrons) is not scaled.

[parameter(1)] is a scale factor used to modify bremsstrahlung production so as to increase the photon population without increasing the number of primary histories. For example, if [parameter(1)] is set equal to two, then there will be twice as much bremsstahlung photon production. The ELECTRON-RR secondary keyword can be used to control the number of secondary electrons generated by this increased population of photons.

[parameter(2)] is the index of the material, according to the order in which the materials are read from the cross section file, on which the impact ionization scaling is based if SCALE-IMPACT is not used. The default is material number 1.

The input zones in which the scaled cross sections are applied must be specified with the BIAS-ZONE keyword. Because the scaling factor must only be set once, this secondary keyword may not be repeated.

(g) **SCALE-EP** *(MITS Only)*

Syntax: SCALE-EP [parameter(1)]

Example: SCALE-EP 2.0

Default: Natural electron-to-photon cross sections are used.

This keyword specifies the factor by which the electron-to-photon cross sections are to be scaled and in which input zones the scaled cross sections are to be used. [parameter(1)] is the scaling factor.

The input zones in which the scaled cross sections are applied must be specified with the BIAS-ZONE keyword. Because the scaling factor must only be set once, this secondary keyword may not be repeated.

(h) **SCALE-IMPACT** *(ITS Only)*

Syntax: SCALE-IMPACT [parameter(1)]

Example: SCALE-IMPACT 20.0

Default: Natural probability of electron impact ionization, except that if the SCALE-BREMS keyword has been used, the scale factors for electron impact ionization will be based on the bremsstrahlung scaling (such that for an electron slowing from the maximum energy to the global electron cutoff energy for every five bremsstrahlung interactions there will be one electron impact ionization event in the material specified by [parameter(2)] of SCALE-BREMS).

[parameter(1)] is used to scale electron impact ionization so as to increase the photon population (line radiation) without increasing the number of primary histories. The cross sections are scaled such that an electron slowing from the maximum energy to the global electron cutoff energy will, on the average, undergo a number of ionization events equal to [parameter(1)] in each material. The values by which the cross sections may be scaled in every material are written to output. The cross sections will never be scaled down (such that fewer ionization events are simulated than with the natural cross sections).

WARNING: Impact ionization scaling is only activated in zones in which SCALE-BREMS is activated.

(i) **SCALE-PE** *(MITS Only)*

Syntax: SCALE-PE [parameter(1)]

Example: SCALE-PE 0.5

Default: Natural photon-to-electron cross sections are used.

This keyword specifies the factor by which the photon-to-electron cross sections are to be scaled and in which input zones the scaled cross sections are to be used. [parameter(1)] is the scaling factor.

The input zones in which the scaled cross sections are applied must be specified with the BIAS-ZONE keyword. Because the scaling factor must only be set once, this secondary keyword may not be repeated.

(j) **TRAP-ELECTRONS** *(Forward Only)*

Syntax: TRAP-ELECTRONS [parameter(1)] [keyword]

Example: TRAP-ELECTRONS 42 CUSTOM-TRAP

Default: The trap-electron energy group is the electron cutoff group.

For ITS, the electron trapping cutoff specified by [parameter(1)] is energy in MeV.

For MITS, the electron trapping cutoff is the lower energy bound of the group specified by [parameter(1)].

Electron trapping is either ineffective or not implemented correctly yet for adjoint calculations.

Electron trapping is not fully functional for CAD. Trapping will only function with "GEOMETRY 3" and "GEOMETRY 4". Trapping will be performed only on subzones that do not include a zone boundary.

The keyword **CUSTOM-TRAP** indicates that customized trapping logic has been included by the user in subroutine SAVE.

5. **BIAS-ZONE**

Syntax: BIAS-ZONE [parameter(1)] [keyword(1)] [keyword(2)] ... [keyword(n)]

Example: BIAS-ZONE 1 RR SCEP

BIAS-ZONE 3 FORCING 0.2 SCPE TRAPE 1

Default: No biasing parameters are activated. Electron trapping is determined by the global parameter.

Selectively turns on the input-zone-dependent bias parameters specified: **ESEC** *(Forward Only)* - electron-produced secondary electrons and scattered photons will be followed in this zone, **FORCING** - forced photon interactions, **PSEC** *(Forward Only)* - photon-produced secondary electrons will be followed in this zone, **RR** *(Forward Only)* - Russian Roulette, **SCBR** *(ITS Only)* - scale bremsstrahlung production of photons, **SCEP** *(MITS Only)* - scale electron-to-photon interactions, **SCPE** *(MITS Only)* - scale photon-to-electron interactions, and/or **TRAPE** - electron trapping.

[parameter(1)] specifies the input zone. This primary keyword may be repeated for each input zone for which it is desired to activate one (or more) of the biasing options.

Additional parameters may be required. After the sub-keyword FORCING, the user must specify the probability of photon interaction in the zone. After the sub-keyword TRAPE, the user must specify the electron trapping cutoff by group in MITS or by energy in ITS.

To use ESEC, the BIAS-GLOBAL/ELECTRAN keyword must also be specified. Here, ESEC specifies exceptions to the global ELECTRAN setting. That is, electron-produced secondary photons and scattered primary photons will be tracked in zones in which BIAS-ZONE/ESEC has been set.

To use PSEC, the BIAS-GLOBAL/PHOTRAN keyword must also be specified. Here, PSEC specifies exceptions to the global no-photon-produced-secondary-electrons setting. That is, photon-produced secondary electrons will be tracked in zones in which BIAS-ZONE/PSEC has been set.

The user cannot simultaneously specify SCEP and SCPE in the same input zone. SCEP and SCPE always refer to scaling the forward cross section, even in adjoint. Thus, when using an SCPE factor greater than one in adjoint, an electron-adjuncton would be more likely to turn into a photon-adjuncton.

TRAPE (zone-dependent electron trapping) does not work well or is not properly understood in adjoint mode.

To use the secondary keywords SCBR, SCEP, or SCPE, the user must specify scaling parameters with the primary keyword BIAS-GLOBAL and its respective secondary keywords.

The keywords BIAS-ZONE and BIASING are alternative methods for setting zone-dependent biasing parameters and may not both be used.

6. **CUTOFF-PHOTONS-ESCAPE** *(Forward Only)*

Syntax: CUTOFF-PHOTONS-ESCAPE

Example: CUTOFF-PHOTONS-ESCAPE

Defaults: Energy of photons below the cutoff energy is deposited locally.

This keyword specifies that photons falling below the cutoff energy are assumed to escape from the problem. A diagnostic in the output states the average energy of photons per history that is assumed to have escaped from the problem.

Use of the ELECTRAN biasing feature causes all electron-produced photons and all scattered photons (except for coherent scattering; see the NO-COHERENT keyword) to be considered below the cutoff energy.

7. **CUTOFFS**

Syntax: CUTOFFS [parameter(1)] - [parameter(6)]

Example: CUTOFFS 0.01 0.1 2 * For ITS

1 3

0.5 0.2

Example: CUTOFFS 48 49 47 2 0 3 * For MITS

1 3

45 45

1 3 4

45 45 40

Defaults: In ITS the global electron cutoff energy equals 5% of the maximum source energy, and the global photon cutoff energy equals 0.01 MeV. In MITS forward mode, the cutoff group is the last group for each species. In MITS adjoint mode, the cutoff group is the first group for each species.

This keyword specifies the global cutoff energy for electrons [parameter(1)] and photons [parameter(2)] (and only in MITS, neutrons [parameter(3)]). In MITS forward mode, the cutoff energy is the lower energy bound of the specified group. In MITS adjoint mode, the cutoff energy is the upper energy bound of the specified group.

This keyword can also be used to specify local cutoff groups. The numbers of input zones for which local cutoff groups are to be specified are given for electrons [parameter(3)] (or in MITS, electrons [parameter(4)], photons [parameter(5)], and neutrons [parameter(6)]). For each of these parameters that are non-zero, two sets of data must follow. The first set contains the input zones for which local cutoff groups are to be specified. The second set specifies the local cutoff energy for the corresponding input zone. The more stringent of the local and global cutoff will be used by the code.

In MITS the indices refer to the local group numbers as they were generated by CEPXS before they were reversed (for adjoint calculations) in the Monte Carlo. In forward mode, electrons which slow down below the lowest energy in the cutoff group are no longer transported and their energy and charge are locally deposited. Photons which downscatter below the cutoff group or which are absorbed (only in the default cutoff group) have their energy locally deposited and transport terminated. In adjoint

mode, particles which speed up beyond the highest energy in the cutoff group will no longer be transported.

WARNING: The CUTOFFS keyword should be used with caution in adjoint mode. Caution is also warranted in using this keyword with neutrons in forward mode. Low energy neutrons may produce high energy photons and may be important in the transport process.

8. **DEPOSITION-UNITS** *(ACCEPT Forward Only)*

Syntax: DEPOSITION-UNITS [keyword] [keyword] [parameter(1)]

Example: DEPOSITION-UNITS MASS SCALE 620.5

'MeV-cm2/g-ph' 'el-cm2/g-ph'

Default: Dose in units of MeV per source particle and charge deposition in units of electrons per source particle.

This keyword modifies the default units of charge and energy deposition outputs for the ACCEPT codes. Three secondary keywords may be used with DEPOSITION-UNITS: **MASS**, **VOLUME**, and **SCALE**. MASS and VOLUME are mutually exclusive. If the MASS keyword is used, deposition values for each input zone will be divided by the mass of material in the zone. If the VOLUME keyword is used, deposition values for each input zone will be divided by the volume of the zone. If the SCALE keyword is used (either separately or in addition to the MASS or VOLUME keyword), deposition values for each input zone will be multiplied by the scaling factor [parameter(1)]. The two character strings on the line following the keyword are the new units for energy and charge deposition that will be used (only) for labels in the output file. The character strings can be up to 15 characters long and should be enclosed in single quotation marks.

WARNING: Accurate MASS or VOLUME scaling depend on the accuracy of the volume data used. Internal calculation of zone volumes is not available for all subzoned bodies or body combinations. Refer to the GEOMETRY keyword for further information.

9. **DETAIL-IONIZE** *(ITS Only)*

Syntax: DETAIL-IONIZE

Example: DETAIL-IONIZE

Defaults: Line radiation is reported, but the source of the ionization is not detailed.

This keyword specifies that line radiation due to electron impact ionization and photon ionization are to be reported separately.

10. **DETECTOR-RESPONSE** *(Adjoint Only)*

Syntax: DETECTOR-RESPONSE

Example: DETECTOR-RESPONSE

Default: No default. User must specify a detector response.

This keyword is the means by which the user specifies what single quantity of interest (known as the detector response in the forward mode) is desired for the adjoint calculation.

(a) **CHARGE**

Syntax: CHARGE

Example: CHARGE

Default: Charge deposition is not the quantity of interest in the adjoint calculation.

This keyword specifies charge deposition for the quantity of interest to be determined in an adjoint calculation. The following MATERIAL sub-keyword must be present.

i. **MATERIAL**

Syntax: MATERIAL [parameter(1)]

Example: MATERIAL 5

Default: No default, this sub-keyword must be present.

This specifies the material in which charge deposition is calculated.

ii. **LOCATION**

Syntax: LOCATION

Example: LOCATION

Default: Point charge deposition calculated at the origin.

See the POSITION keyword for secondary keywords. For now, normalization logic is only included for POINT and VOLUME distributions.

(b) **DOSE**

Syntax: DOSE

Example: DOSE

Default: Energy deposition is not the quantity of interest in the adjoint calculation.

This keyword specifies energy deposition for the quantity of interest to be determined in an adjoint calculation. The following MATERIAL sub-keyword must be present.

i. **MATERIAL**

Syntax: MATERIAL [parameter(1)]

Example: MATERIAL 5

Default: No default, this sub-keyword must be present.

This specifies the material in which energy deposition is calculated.

ii. **LOCATION**

Syntax: LOCATION

Example: LOCATION

Default: Point dose deposition calculated at the origin.

See the POSITION keyword for secondary keywords.For now, normalization logic is only included for POINT and VOLUME distributions.

(c) **ESCAPE**

Syntax: ESCAPE [keyword]

Example: ESCAPE PHOTONS

Default: No default for the keyword ESCAPE or its tertiary keyword.

This keyword specifies particle escape (or leakage) for the quantity of interest to be determined in an adjoint calculation. The tertiary keyword must be present and be either **ELECTRONS**, **PHOTONS**, or **NEUTRONS** for electron-escape, photon-escape, or neutron-escape, respectively.

The user can further specify the type of escaping quantity through the following sub-keywords:

i. **GROUP**

Syntax: GROUP [parameter(1)]

Example: GROUP 8

Default: The particle escape is integrated over all energy groups.

This sub-keyword specifies the group index of the quantity of interest. The order of the group structure is that produced by CEPXS before the inversion which occurs in the Monte Carlo in adjoint mode.

ii. **LOCATION**

Syntax: LOCATION

Example: LOCATION

Default: No default surface for particle escape.

See the POSITION keyword for secondary keywords. Only the SURFACE keywords are functional. The sub-keyword SURFACE specifies the surface through which forward escape detector-response will be calculated.

iii. **BINT**

Syntax: BINT [parameter(1)] [parameter(2)]

Example: BINT 30.0 45.0

Default: The particle escape is integrated over lab angles from 0-90 degrees.

For now, the reference direction is LOCAL-NORMAL to the specified escape surface. The escape distribution is between angles given by [parameter(1)] and [parameter(2)] with defaults of 0 and 90 degrees, respectively. The escape direction distribution bin is based on a cosine-law to yield particle current.

(d) **KERMA**

Syntax: KERMA

Example: KERMA

Default: KERMA is not the quantity of interest in the adjoint calculation.

This keyword specifies KERMA (Kinetic Energy Released in MAterial) for the quantity of interest to be determined in an adjoint calculation. Operationally, the kerma dose is calculated from the photon flux. Photon-generated electrons are assumed locally deposited with a small crude correction for escaping bremsstrahlung. The following MATERIAL sub-keyword must be present.

i. **MATERIAL**

Syntax: MATERIAL [parameter(1)]

Example: MATERIAL 5

Default: No default, this sub-keyword must be present.

This specifies the material in which KERMA is calculated.

ii. **LOCATION**

Syntax: LOCATION

Example: LOCATION

Default: Point KERMA calculated at the origin.

See the POSITION keyword for secondary keywords. For now, normalization logic is only included for POINT and VOLUME distributions.

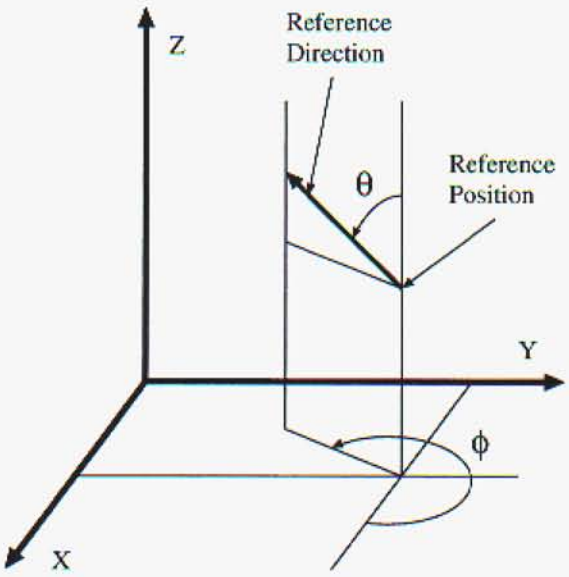


Figure 1. Source position and reference direction

11. **DIRECTION** *(Forward Only)*

Syntax: DIRECTION [parameter(1)] [parameter(2)]

Example: DIRECTION 90.0 90.0

Default: reference direction is positive-z direction.

This keyword is used to define the source reference direction and the distribution of particles in angle relative to the reference direction (either isotropic or cosine-law).

[parameter(1)] is the spherical polar angle θ, in degrees, and [parameter(2)] *(CYLTRAN and ACCEPT Only)* is the azimuthal angle ϕthat define the reference direction.

The meaning of source parameters specified with the DIRECTION and POSITION keywords is illustrated in Figure 1.

(a) **ISOTROPIC**

Syntax: ISOTROPIC [parameter(1)] [parameter(2)]

Example: ISOTROPIC 10.0 45.0

Default: Monodirectional in the reference direction.

Defines the distribution of source particles as isotropic between angles given by [parameter(1)] and [parameter(2)], with respect to the reference direction. The default values for [parameter(1)] and [parameter(2)] are 0 and 90 degrees, respectively.

(b) **COSINE-LAW**

Syntax: COSINE-LAW [parameter(1)] [parameter(2)]

Example: COSINE-LAW

Default: Monodirectional in the reference direction.

Defines the distribution of source particles as proportional to the cosine of the angle with respect to the reference direction. The source is distributed between angles given by [parameter(1)] and [parameter(2)] with defaults of 0 and 90 degrees, respectively.

12. **DUMP**

Syntax: DUMP

Example: DUMP

Default: no dump

If the DUMP keyword is present, a dump file will be written after each batch to "fort.10". If the dump file is to be used for a subsequent restart (see primary keyword RESTART), it must be saved.

13. **ECHO**

Syntax: ECHO [parameter(1)]

Example: ECHO 1

Default: no echo

If "ECHO 1" is inserted in the input stream, all subsequent input card images will be echoed to the Fortran unit 6 output. If "ECHO 0" is inserted in the input stream, subsequent input card images will not be echoed.

14. **ELECTRONS** *(Forward Only)*

Syntax: ELECTRONS

Example: ELECTRONS

Default: electron source if neither PHOTONS or NEUTRONS primary keyword is used.

This keyword defines the source particles to be electrons rather than photons.

Electrons currently can not be transported in CAD geometries in continuous-energy ITS.

15. **ELECTRON-EMISSION** *(ITS ACCEPT Forward Only)*

Syntax: ELECTRON-EMISSION [parameter(1)] SURFACES [parameter(2)]
SURFACE [parameter(3)] BODY [parameter(4)]
[parameter(5)] [parameter(6)] [keyword] [parameter(7)]

Example: ELECTRON-EMISSION 3 SURFACES 2
SURFACE 1 BODY 4
3 3
SURFACE 2 BODY 5
2 1 AZ 8

Default: Electron emission is not tallied.

This keyword signals that electron emission is to be tallied. If [parameter(1)] is positive, all electrons entering the zone number specified will be terminated. If [parameter(1)] is negative, all electrons entering the zone number specified (by the absolute value of the parameter) will continue to be transported in the zone. Electron emission tallies will be made if they passed through a specified surface to exit the zone they were in prior to entering the electron-emission zone. **WARNING:** In many cases the two adjacent zones will share a surface, so that the surface that is passed through to exit the prior zone will be the same as the surface passed through to enter the electron-emission zone, but this is not always the case so the user is encouraged to give this point careful consideration.

The number of surfaces on which tallies are desired is specified with [parameter(2)]. This must be followed immediately with 2×[parameter(2)] lines. The user must specify the surface indices of the body numbers for which tallies are desired using [parameter(3)] and [parameter(4)], respectively. Figure 2 illustrates how many of the body types used in ACCEPT have their surfaces numbered. The rectangular parallelepiped (RPP) is numbered the same as the BOX where A1 points in the positive x-direction, A2 points in the positive y-direction, and A3 points in the negative z-direction. The right circular cylinder (RCC) is numbered the same as the TRC.

The subsurfacing (equidistant subdivision) of each surface must be specified using [parameter(5)] and [parameter(6)] to request the number of divisions in the two dimensions. The meaning of these "U" and "V" parameters depend upon the surface being subsurfaced and are discussed below. The output will refer only to a single subsurface number, corresponding to incrementing the [parameter(6)] value first. For example, the 6th subsurface in a "3 3" subsurfacing corresponds to the 2nd U-division and the 3rd V-division.

To specify such a reference direction, the input line with the two subsurfacing integers may contain the keyword **AZ**, followed by any body number except an RPP. The zero reference vector is the component of the V vector of the **AZ** body that is perpendicular to the H vector of the subsurfaced body. If that definition of the reference vector is null, the code attempts to use the +i, +j, and +k vector for the zero azimuth vector, in that order. If no **AZ** body is specified, the +i vector will be used as the body vector. This keyword may be applied to all surfaces of a TRC, RCC, and TOR. For the planar surfaces of a TRC or RCC, the reference direction also defines the zero direction for any angular division specified with the NBINP sub-keyword.

For binning in energy and angle, the same secondary keywords apply as for ELECTRON-ESCAPE (NBINE, NBINT, NBINP). Directions for electron emission tallies are relative to the local surface normal. That is, zero degrees theta is the normal into the emission zone. Zero degrees phi is defined differently depending upon the surface. The phi coordinate is based on a right-hand rule of zero theta and zero phi.

This feature is currently available for the surfaces of the following bodies:

(a) RPP – The U-V surfaces are determined as follows: for constant-x planes, they are the y and z divisions; for constant-y planes, they are the z and x divisions; for constant-z planes, they are the x and y divisions, respectively. Zero degrees phi is defined by the U coordinate axis.

(b) BOX – The U-V surfaces are determined as follows: for constant-$\mathbf{A}_1$ planes, they are the $\mathbf{A}_2$ and $\mathbf{A}_3$ divisions; for constant-$\mathbf{A}_2$ divisions, they are the $\mathbf{A}_3$ and $\mathbf{A}_1$ divisions; for constant-$\mathbf{A}_3$, they are the $\mathbf{A}_1$ and $\mathbf{A}_2$ divisions, respectively. Zero degrees phi is defined by the U coordinate axis.

(c) WED – The U-V surfaces are determined in the same manner as for the BOX, except there is no surface 4 and surface 2 is uses the vector $\mathbf{A}_1$-$\mathbf{A}_2$ for U divisions. The rectangular divisions of surfaces 5 and 6 are not modified for the triangular surfaces of the wedge, so only half of the divisions may be tallied. Zero degrees phi is defined by the U coordinate axis.

(d) RCC – On the planar surfaces 1 and 2, the U-V divisions are in angle about the axis of the cylinder and in radius, respectively. On the cylindrical surface, the U-V divisions are in angle about the axis of the cylinder and in the axial coordinate (or height) of the cylinder, respectively. The zero phi direction on the planar surfaces is determined by the zero angle about the axis, which is determined by the **AZ** logic. The zero phi direction on the cylindrical surface is in the direction of the **H** vector of the body.

(e) TRC – The U-V divisions are determined in the same manner as for the RCC, except that the zero phi direction on the conical surface is in the direction of the point of the cone. Note that the direction to the point of the cone depends upon the respective radii of the bases of the TRC, rather than on the **H** vector of the body.

(f) SPH – The U-V divisions are azimuthal about the laboratory z axis with respective to the postive-x axis and in polar angle with respect to the positive-z axis. The zero phi direction is tangential to the surface of the sphere in the direction of the positive-z pole of the body.

(g) TOR – The U-V divisions are the poloidal-phi and toroidal-phi directions. This is analogous to the division of the cylindrical surface of the RCC, where the torus is a cylinder with its axis wrapped in a circle. Thus, the RCC azimuthal angle about the axis is the TOR poloidal angle, and the RCC axial coordinate is the TOR toroidal angle. The zero phi direction is tangential to the surface of the torus, such that it is in the same direction as the **H** vector of the torus on the outside and in the opposite direction of the **H** vector of the torus on the inside.

16. **ELECTRON-ESCAPE** *(Forward Only)*

Syntax: ELECTRON-ESCAPE

Example: ELECTRON-ESCAPE

Default: Electron escape not tallied.

This keyword signals that electron escape is to be tallied. The following are secondary keywords associated with this primary keyword that describe the bin structure used in tallying electron escape.

(a) **NBINE**

Syntax: NBINE [parameter(1)] [keyword]

Example: NBINE 5 USER

10 20 30 40 50

Default: In ITS, 10 bins of equal width are used. In MITS, the bin structure corresponds to the electron group structure.

[parameter(1)] is the number of energy bins.

In ITS, choices for [keyword] are:

i. **LOG** – Logarithmic grid spacing, with some parameter x such that $E_{cutoff} = x^{[parameter(1)]}E_0$, where E_{cutoff} is the cutoff energy and E_0 is the source energy, and the grid values are defined as $E_{i+1} = xE_i$.

ii. **USER** – User defined energy grid. The code will then read lower bound energies (MeV) for the number of energy bins specified by [parameter(1)] in descending order. The maximum lower bound must be less than the maximum source energy.

For either of these tertiary keywords, the user must insure that the energy is less than or equal to the global electron cutoff (if less than, the grid will be truncated).

In MITS, this keyword allows the user to "collapse" the default bin structure (which is one bin for every group in the electron group structure) into fewer bins.

i. **USER** – The USER secondary keyword **must** be present, and this line must be followed by [parameter(1)] numbers (integers) which specify group indices. The lower-bound energies of the specified groups form the lower-bound energies of the escape bins. The indices should appear in strictly increasing order.

(b) **NBINT**

Syntax: NBINT [parameter(1)] [keyword]

Example: NBINT 5 USER

10.0 30.0 90.0 135.0 180.0

Default: 18 bins of 10 degrees each up to 180 degrees.

This keyword allows the user to define a polar bin structure for recording electron escape. [parameter(1)] specifies the number of polar bins.

If the **USER** secondary keyword is present, then this line must be followed by [parameter(1)] numbers which specify angle bins in ascending order up to 180 degrees.

If the **DIRECTION-SPACE** *(CYLTRAN and ACCEPT Only)* keyword is used, [parameter(1)] polar bins are internally generated and azimuthal bins are generated such that angular bins are approximately equal in size. See the warning under sub-keyword NBINP.

If no secondary keyword is present, then [parameter(1)] equal width bins will be defined.

(c) **NBINP** *(CYLTRAN and ACCEPT Only)*

Syntax: NBINP [parameter(1)] [USER]

Example: NBINP 5 USER

10.0 30.0 140.0 235.0 360.0

Default: 1 bin of 360 degrees.

This keyword allows the user to define an azimuthal bin structure for recording electron escape. [parameter(1)] specifies the number of azimuthal bins. If the **USER** secondary keyword is present, then this line must be followed by [parameter(1)] numbers which specify angle bins in ascending order up to 360 degrees. If the USER keyword is not present, then [parameter(1)] equal width bins will be defined. Note: NBINP cannot be used with the NBINT/DIRECTION-SPACE option.

WARNING: Because azimuthal scoring of escaping radiation is only rarely requested, the azimuthal dimension of the escape arrays has been suppressed to the default value of one bin in order to save memory. Consequently, if the keyword NBINP is used to obtain more than one azimuthal bin, the user must increase the dimension of the azimuthal arrays by increasing the value of the parameter IKMAZ (IKPMAZ for photon escape, IKNMAZ for neutron escape) as defined in the params.h file.

17. **ELECTRON-FLUX** *(Forward Only)*

Syntax: ELECTRON-FLUX [parameter(1)] [parameter(2)]

Example: ELECTRON-FLUX 3 5

Default: No electron flux tallied.

This keyword signals that electron flux is to be tallied in all subzones for input zones [parameter(1)] through [parameter(2)]. The automatic subzoning features of the ITS codes are discussed in more detail in the Subzoning section. If either parameter is omitted or 0, flux will be calculated in all zones. The same secondary keywords apply as for ELECTRON-ESCAPE (NBINE, NBINT, NBINP).

Calculation of electron flux in zones where macroscopic fields have been specified is not allowed. The user must insure that the zone dependent electron cutoff energies (see keyword CUTOFFS) for zones [parameter(1)] through [parameter(2)] are all equal.

18. **ELECTRON-SURFACE-SOURCE** *(Adjoint Only)*

Syntax: ELECTRON-SURFACE-SOURCE

Example: ELECTRON-SURFACE-SOURCE

Default: No forward electron surface sources in adjoint mode.

This keyword signals that electron escape is to be tallied. The following keywords are secondary keywords associated with this primary keyword that describe the bin structure used in tallying electron escape. Directions are in the LOCAL-NORMAL frame unless the DELTA0-AVE sub-keyword is used.

(a) **NBINE**

Syntax: NBINE [parameter(1)] USER

Example: NBINE 5 USER
1 11 21 31 41

Default: The bin structure corresponds to the electron group structure.

This keyword allows the user to "collapse" the default bin structure (which is one bin for every group in the electron group structure) into fewer bins. [parameter(1)] specifies the number of energy bins. The **USER** secondary keyword must be present, and this line must be followed by [parameter(1)] numbers (integers) which specify group indices (before inversion, for adjoint). In adjoint mode, the upper-bound energies of the specified groups form the upper-bound energies of the escape bins. The indices should appear in strictly increasing order.

(b) **NBINT**

Syntax: NBINT [parameter(1)] [keyword] [keyword]

Example: NBINT 6 USER COSINE-LAW
5. 10.5 22.25 45 70 90

Default: Nine cosine-law sources within one of nine equal polar-angle bins with azimuthal symmetry about the z-axis.

This keyword allows the user to specify the number, angular-extent, and type of angular distribution of forward sources for an adjoint calculation with electron surface sources. [parameter(1)] specifies the number of source polar-angle bins. Without either the USER or DIRECTION-SPACE keyword, this will generate [parameter(1)] equal solid-angle polar-angle bins. When the **USER** keyword is used, the following line must contain [parameter(1)] values which are the upper bounds of the polar-angle bins. When the **DIRECTION-SPACE** *(CYLTRAN and ACCEPT Only)* keyword is used , [parameter(1)] polar bins are internally generated and azimuthal bins are generated such that angular bins are approximately equal in size. DIRECTION-SPACE can only be used with the DELTA0-AVE setting.

The final keyword describes the source angular distribution within these bins. It should be one of the following (COSINE-LAW is the default):

i. **ISOTROPIC** The forward-source is uniform in angle within the specified angular bin, e.g. a thin surface of a radioactive material.

ii. **COSINE-LAW** The forward source has a cosine-law distribution with respect to the surface normal, e.g., this corresponds to the "isotropic-flux" sources of cosmic particles.

iii. **DELTA0-AVE** The forward source is the average, within the specified angular bin, of all plane-wave sources with normals within the specified angular bin. For each plane wave, the source is a delta function normal to the plane, $\delta(\Omega - 1)$. This might be used to help assess vulnerability to plane-wave sources.

(c) **NBINP** *(CYLTRAN and ACCEPT Only)*

Syntax: NBINP [parameter(1)] [USER]

Example: NBINP 5 USER
10.0 30.0 140.0 235.0 360.0

Default: 1 bin of 360 degrees.

This keyword allows the user to define an azimuthal bin structure for recording angular distribution of forward sources for an adjoint calculation with electron surface sources. [parameter(1)] specifies the number of azimuthal bins. If the **USER** secondary keyword is present, then this line must be followed by [parameter(1)] numbers which specify angle bins in ascending order up to 360 degrees. If the USER keyword is not present, then [parameter(1)] equal width bins will be defined. NBINP cannot be used with the NBINT/DIRECTION-SPACE option.

WARNING: Because azimuthal scoring of escaping radiation is only rarely requested, the azimuthal dimension of the escape arrays has been suppressed to the default value of one bin in order to save memory. Consequently, if the keyword NBINP is used to obtain more than one azimuthal bin, the user must increase the dimension of the azimuthal arrays by increasing the value of the parameter IKMAZ (IKPMAZ for photon escape, IKNMAZ for neutron escape) as defined in the params.h file.

19. **ELECTRON-VOLUME-SOURCE** *(Adjoint Only)*

Syntax: ELECTRON-VOLUME-SOURCE [parameter(1)] [parameter(2)]

Example: ELECTRON-VOLUME-SOURCE 5 7

Default: No volume source.

Not functional. If it were functional:

This keyword signals that a forward source (i.e., the flux of adjunctons) is to be tallied in all subzones for input zones [parameter(1)] through [parameter(2)]. If either parameter is omitted or 0, the source will be calculated in all zones. The same secondary keywords apply as for ELECTRON-ESCAPE (NBINE, NBINT, NBINP).

20. **ENERGY** *(Forward Only)*

Syntax: ENERGY [parameter(1)] [keyword] [parameter(2)]

Example: ENERGY 4

Example: ENERGY GROUP 1 (MITS Only)

Default: For ITS, 1.0 MeV monoenergetic source. For MITS, a mono-group source in the highest energy group.

In ITS, the source is specified as a mono-energetic source with energy [parameter(1)] in MeV.

In MITS, the source may be specified as a mono-energetic source or as a mono-group source. For an electron source, the user may specify a monoenergetic source with the energy specified by [parameter(1)] in MeV. For any particle species, if the keyword GROUP appears on the line, then the associated [parameter(2)] specifies the single group index which will be used for all the source particles. The specific energy of the individual histories will be sampled uniformly over the width of the group. Thus, if the user desires a mono-energetic source, the cross sections generated by CEPXS must include the appropriate single-energy (or "source-line") group.

21. **ESCAPE-SURFACES** *(Forward Only)*

Syntax: ESCAPE-SURFACES [parameter(1)]

Example: ESCAPE-SURFACES 3

Default: All surfaces of the escape zone (tallied as a single surface) for ACCEPT. Both surfaces (ZMIN and ZMAX) for TIGER. All three surfaces (ZMAX, ZMIN, and RMAX) for CYLTRAN.

This specifies the number of escape surfaces for which the integral particle escape or the requested particle escape will be displayed. This keyword must be followed by [parameter(1)] separate lines of the secondary keyword SURFACE.

Any specification for CAD will be ignored. All escaping particles are scored in a single body/surface tally.

(a) **SURFACE**

Syntax: SURFACE [keyword]	(non-ACCEPT codes)
or SURFACE [parameter(1)] BODY [parameter(2)]	(ACCEPT codes)
Example: SURFACE ZMAX	(non-ACCEPT codes)
or SURFACE 3 BODY 2	(ACCEPT codes)

This sub-keyword specifies the surface index through which the escaping particles will be calculated.

For TIGER and CYLTRAN, surface **ZMIN** refers to the minimum-z surface and surface **ZMAX** is the maximum-z surface. For CYLTRAN, surface **RMAX** is the lateral escape surface at maximum radius. For ACCEPT, it is necessary to specify both the surface index and the **BODY** number. Figure 2 illustrates how many of the body types used in ACCEPT have their surfaces numbered. In the case of the arbitrary polyhedron (ARB), the user explicitly specifies the order in the input. The right circular cylinder is numbered the same as the truncated right-circular cone (TRC). The rectangular parallelepiped (RPP) is numbered the same as the BOX where A1 points in the positive x-direction, A2 points in the positive y-direction, and A3 points in the positive z-direction.

The user is cautioned to make sure that the desired surface has a unique description for the way the user has specified the geometry, otherwise the result may be invalid. For example, if a zone is defiined as the union of two bodies and those two bodies have coincidental surfaces, a particle may exit the zone surface through either one of the two body surfaces.

22. **FILE-NAMES**

Syntax: FILE-NAMES

Example: FILE-NAMES

Default: Default names (based on Fortran unit names) are used.

This keyword allows the user to specify names of files to be opened for input and output. File names are only allowed to be up to 16 characters.

(a) **DUMP-FILE**

Syntax: DUMP-FILE
[keyword]
Example: DUMP-FILE
Dump
Default: Dump data needed for a restart is written to "fort.10".

This keyword specifies that dump data will be written to the file named [keyword].

(b) **INTERMEDIATE-FILE**

Syntax: INTERMEDIATE-FILE
[keyword]
Example: INTERMEDIATE-FILE
InterOut
Default: Intermediate output is written to "fort.12".

This keyword specifies that intermediate output will be written to the file named [keyword].

(c) **FINITE-ELEMENT-FILE** *(ACCEPT Forward Only)*

Syntax: FINITE-ELEMENT-FILE
[keyword]
Example: FINITE-ELEMENT-FILE
torus.dat
Default: Finite element data is written to "fort.3".

This keyword specifies that finite-element output will be written to the file named [keyword].

(d) **RESTART-FILE**

Syntax: RESTART-FILE
[keyword]
Example: RESTART-FILE
Restart
Default: Restart data is read from "fort.14".

This keyword specifies that restart data will be read from the file named [keyword].

(e) **XSECTION-FILE**

Syntax: XSECTION-FILE
[keyword]
Example: XSECTION-FILE
XSfile
Default: Cross sections are read from "fort.11".

This keyword specifies that cross sections will be read from the file named [keyword].

23. **FINITE-ELEMENT-FORMAT** *(ACCEPT Forward Only)*

Syntax: FINITE-ELEMENT-FORMAT [CHARGE] [DOSE]

Example: FINITE-ELEMENT-FORMAT CHARGE

Default: no finite-element file

This keyword signals the creation of a Tecplot® [32] finite-element-format file written to "fort.3". The file contains charge deposition (secondary keyword CHARGE) and/or energy deposition (secondary keyword DOSE) for each subzone. At least one secondary keyword is required. Each ITS zone that is subzoned is written as a separate Tecplot zone. Each subzone of an ITS zone represents an element in the Tecplot zone. The energy and charge deposition data are written as volume-averaged values associated with each element. Since Tecplot expects nodal values, each element node has energy and/or charge depostion values but only the first node of each element represents the correct volume-averaged value for that element.

24. **GEOMETRY**

This keyword signals the beginning of the geometry information. The choice from among the following usages depends on which of the member codes of ITS has been selected: TIGER, CYLTRAN, ACCEPT, or CAD. (There are also minor differences between the continuous-energy and multigroup codesin the biasing settings.) For a more detailed discussion of the GEOMETRY keyword input, the user is referred to the geometry documentation in the sections specific to the member code chosen. The ACCEPT Geometry section is also relevant to CAD.

- *(TIGER codes)*

Syntax: GEOMETRY [parameter(1)]
[parameter(2)] [parameter(3)] [parameter(4)]

Example: GEOMETRY 3
3 1 0.1
1 10 12.0
2 5 0.15

Default: no default

[parameter(1)] is the number of input layers. Immediately after the keyword line there must follow a series of [parameter(1)] lines, one for each layer, containing [parameters(2)] through [parameter(4)] which specify the material, the number of subzones, and the layer thickness in cm.

- *(CYLTRAN codes)*

Syntax: GEOMETRY [parameter(1)]
[parameter(2)] [parameter(3)] ... [parameter(10)]

Example: GEOMETRY 3
-0.5 0.20 0.0 10.5 3 1 1 5 0

0.2 12.45 0.0 10.5 1 0 0 0 1

-0.5 12.45 10.5 12.05 2

Default: no default

[parameter(1)] is the number of input zones. Immediately after the keyword line there must follow a series of [parameter(1)] lines, one for each input zone, containing [parameters(2)] through [parameter(6)]. These parameters specify the minimum z boundary, the maximum z boundary, the minimum ρ boundary, the maximum ρ boundary, the material, the number of ϕ subzones, the number of ρ subzones, and the number of z subzones. In the case of the MCODES, the macroscopic field flag is inserted as [parameter(10)]. This flag specifies the macroscopic fields that are present in the given zone and may have the values: 0 for no field, 1 for magnetic field only, and 2 for electric field (and also possibly magnetic field). All boundaries are given in cm. When the fields trailing the material index are left blank, no subzoning is imposed and there will be no fields in the given zone.

- *(ACCEPT codes)*

Syntax: GEOMETRY [parameter(1)] [parameter(2)] [SUBZONE-ONLY]

(Input Body Descriptions)

END

(Input Zone Descriptions)

END

(Subzoning Descriptions)

END

[parameter(3)] [parameter(4)]

Example: GEOMETRY 0 1

RCC 0.0 0.0 0.0 0.0 0.0 1.0 0.25

RCC 0.0 0.0 0.0 0.0 0.0 1.0 0.5

SPH 0.0 0.0 0.5 2.0

SPH 0.0 0.0 0.5 3.0

END

Z1 +1

Z2 +2 –1

Z3 +3 –2

Z4 +4 –3

Z5 –4

END

SUBZONE 1

1 10 10

SUBZONE 2

1 10 10 RCC-RCC AZ 4

END

3
1
2
1
0

Default: no default

This keyword signals the beginning of geometry input for the ACCEPT codes. The value of [parameter(1)] determines the option for setting the subzone volumes in cm^3:

(a) 0 (normal and default) causes the volumes to be set internally to 1.0.

(b) 1 causes the code to read volumes from the input stream.

(c) 2 requires that the user provide the necessary logic for computing the volumes at the appropriate place in subroutine VOLACC.

(d) 3 causes volumes of all subzones to be calculated, while the volumes of input zones that are not subzoned are set to 1.0 cm^3.

(e) –3 has the same effect as 3, but causes printing of volumes to be suppressed.

(f) 4 has the same effect as 3, but subzone volumes for user specified zones may be overwritten.

Tracking debug is turned off or on according to whether [parameter(2)] is set equal to 0 (default) or not, respectively. **WARNING:** The tracking debug feature may produce a lot of information. It is suggested that a single history be simulated to assess the size of the resulting output file produced.

The SUBZONE-ONLY sub-keyword causes ACCEPT to process input through the geometry, write a FINITE-ELEMENT-FORMAT file, and then quit. No further processing of input is performed and no particle transport is performed. This feature is used to produce a refined subzone structure without energy or charge deposition results. The resulting file can be used when mapping the results of a previous calculation performed on a coarse subzone structure onto this refined subzone structure. There is a tool to perform this mapping called MAPPER.

What follows the keyword line is the list of primitive bodies used to construct the combinatorial geometry followed by an END line, the list of input zones constructed from the primitive bodies followed by an END line, the list of subzoning specifications followed by an END line, and possibly a list of zone volumes depending upon the setting of [parameter(1)].

Immediately after this information, there must follow a series of lines, one for each input zone, with each containing [parameter(3)]. These parameters specify the material for each input zone. In the case of the MCODES, the macroscopic field flag is inserted as [parameter(4)] on each line. This flag specifies the macroscopic fields that are present in the given zone and may have the values: 0 for no field, 1 for magnetic field only, and 2 for electric field (and also possibly magnetic field).

- *(CAD codes)*

  The format of geometry keyword for CAD is the same as for the ACCEPT codes. However, there are different requirements regarding what information must be present and differences in how the information is used.

  Bodies must be specified if the code is not running in ACIS_ONLY mode. In ACIS_ONLY mode, CG bodies may be used to specify the spatial distribution of a source.

  Zones are required if the code is not running in ACIS_ONLY mode. If zones are present in ACIS_ONLY mode, they are ignored. For mirroring, the zones must be identical to the CAD description in both geometry and zone numbering.

Although bodies and zones are optional in ACIS_ONLY mode, the END statements must be included even if the sections are otherwise empty.

In HYBRID mode, CG zones may be superimposed on the CAD geometry. The escape zone must be specified as the last zone as the complement of an RPP (that is, everything outside of a body that is an RPP). The RPP should be equal to or larger than the CAD assembly bounding box. A "defined" void (corresponding to a CAD "undefined" void) must be specified as the second to last zone. This should be the RPP (used to specify the escape zone) minus all other defined CG space.

Materials must be specified for zones in all modes. In ACIS_ONLY mode, materials should be assigned to all CAD zones, and two additional void assignments should be made for the undefined void and the escape zone. For HYBRID calculations, material assignments should be made for all CAD zones followed by material assignments for all CG zones. Since the undefined void and escape zone are included in the CG description, no additional assignments are required for these.

25. **HISTORIES**

Syntax: HISTORIES [parameter(1)]

Example: HISTORIES 100000

Default: 1000 histories

This specifies the total number of primary particle histories to be followed. [parameter(1)] cannot be greater than 2,147,483,647 (i.e., 2^{31}-1). To simulate more histories, the HISTORIES-PER-BATCH keyword should be used. HISTORIES and HISTORIES-PER-BATCH are mutually exclusive keywords.

26. **HISTORIES-PER-BATCH**

Syntax: HISTORIES-PER-BATCH [parameter(1)]

Example: HISTORIES-PER-BATCH 10000

Default: 1000 total histories

This specifies the number of primary particle histories to be followed per batch. [parameter(1)] cannot be greater than 2,147,483,647 (i.e., 2^{31}-1). To simulate more histories, the number of batches should be increased. HISTORIES and HISTORIES-PER-BATCH are mutually exclusive keywords.

27. **MICRO** *(MITS Forward Only)*

Syntax: MICRO

Example: MICRO

Default: Energy and charge deposition is determined by folding the flux into the appropriate cross section (e.g., restricted stopping power for electron energy deposition), with an additional microscopic contribution for particles that fall below cutoff.

With this keyword, energy and charge deposition are entirely microscopic. This has been observed to be more efficient for calculating charge deposition, but less efficient for calculating energy deposition.

28. **NEUTRONS** *(MITS Forward Only)*

Syntax: NEUTRONS

Example: NEUTRONS

Default: electron source

This keyword defines the source particles to be neutrons rather than electrons.

29. **NEUTRON-ESCAPE** *(MITS Forward Only)*

See ELECTRON-ESCAPE

30. **NEUTRON-FLUX** *(MITS Forward Only)*

See ELECTRON-FLUX

31. **NEUTRON-SURFACE-SOURCE** *(Adjoint Only)*

See ELECTRON-SURFACE-SOURCE

32. **NEUTRON-VOLUME-SOURCE** *(Adjoint Only)*

See ELECTRON-VOLUME-SOURCE

33. **NEW-DATA-SET**

Syntax: NEW-DATA-SET

Example: NEW-DATA-SET

Default: one run.

This keyword signifies that the data set for a particular Monte Carlo run has been read and that the data set for a new Monte Carlo run follows. Its purpose is to permit multiple Monte Carlo runs within a single code execution. **Its usage represents the exception to the rule that the primary keywords are order independent.**

The cross section file used for the calculation must contain the cross section data necessary for running all of the problems. The input data being described in the present section must be repeated for each problem, and the input data sets for the different problems must be separated from one another by a line containing this keyword.

For CAD calculations, the prmfile and SAT files will not be reread for each run. Therefore, all of the problems must use the same CAD calculation parameters and the same CAD geometry files.

34. **NO-COHERENT** *(ITS Only)*

Syntax: NO-COHERENT

Example: NO-COHERENT

Default: Coherent photon scattering will be included in the calculation.

This keyword deactivates the simulation of coherent photon scattering.

35. **NO-DEPOSITION-OUTPUT** *(Forward Only)*

Syntax: NO-DEPOSITION-OUTPUT

Example: NO-DEPOSITION-OUTPUT

Default: Energy and charge deposition output is written to "fort.12" and unit 6 output.

This keyword supresses energy and charge deposition output. This may prove useful when the FINITE-ELEMENT-FORMAT keyword is used with numerous subzones.

36. **NO-INCOH-BINDING** *(ITS Only)*

Syntax: NO-INCOH-BINDING

Example: NO-INCOH-BINDING

Default: Incoherent photon scattering will include binding effects.

This keyword causes incoherent photon scattering to be simulated in the Klein-Nishina or free-electron approximation.

37. **NO-INTERMEDIATE-OUTPUT**

Syntax: NO-INTERMEDIATE-OUTPUT

Example: NO-INTERMEDIATE-OUTPUT

Default: Intermediate output is written to "fort.12".

This keyword specifies that output will only be written upon completion of the calculation.

38. **NO-KICKING** *(ITS Only)*

Syntax: NO-KICKING

Example: NO-KICKING

Defaults: Terminal processing of electrons and positrons includes kicking, except in the MCODES.

This keyword is intended for development purposes only. The "kicking" of electrons and positrons is an approximation that moves the particle to account for transport at lower energies. This is based on the remaining practical range of the particle. Using this keyword will cause the particle energy and charge to be locally deposited. In the MCODES, there is no kicking, and using this keyword is redundant.

39. **NO-KNOCKONS** *(ITS Only)*

Syntax: NO-KNOCKONS

Example: NO-KNOCKONS

Defaults: Secondary knock-on electrons are produced.

This keyword is intended for development purposes only. Production of secondary knock-on electrons is disabled, however primary electron energy loss and energy loss staggling are not affected. See the NO-STRAGGLING keyword.

40. **NO-STRAGGLING** *(ITS Only)*

Syntax: NO-STRAGGLING

Example: NO-STRAGGLING

Defaults: Energy loss straggling is applied to electrons.

This keyword is intended for development purposes only. Energy loss straggling is disabled for all electron transport.

41. **NO-SZDEPOSITION-OUTPUT** *(Forward Only)*

Syntax: NO-SZDEPOSITION-OUTPUT

Example: NO-SZDEPOSITION-OUTPUT

Default: Energy and charge deposition output is written to "fort.12" and unit 6 output for all zones and subzones.

This keyword supresses energy and charge deposition output for all subzones. This may prove useful when the FINITE-ELEMENT-FORMAT keyword is used with numerous subzones. Unlike the NO-DEPOSITION-OUTPUT keyword, this keyword will allow deposition quantities for zones to be printed out. If the NO-DEPOSITION-OUTPUT keyword is used, this keyword has no effect.

42. **PHOTONS** *(Forward Only)*

Syntax: PHOTONS [parameter(1)]

Example: PHOTONS 1

Default: electron source

This keyword defines the source particles to be photons rather than electrons.

For ITS only, if [parameter(1)] is 0 or omitted, then unscattered photons will be excluded from photon flux and escape scores. Otherwise, unscattered photons will be included in photon flux and escape scores.

43. **PHOTON-ESCAPE** *(Forward Only)*

See ELECTRON-ESCAPE

44. **PHOTON-FLUX** *(Forward Only)*

See ELECTRON-FLUX

45. **PHOTON-SURFACE-SOURCE** *(Adjoint Only)*

See ELECTRON-SURFACE-SOURCE with the following addition.

For the PHOTON-SURFACE-SOURCE keyword, the DIRECTION-SPACE secondary keyword has the following optional secondary keywords.

Syntax: . . . DIRECTION-SPACE [keyword][keyword][keyword] [parameter(1)]

Example: NBINT 6 DIRECTION-SPACE DELTA0-AVE RAYTRACE RAYPRINT 2
50 90.1 75.5 270

Default: No ray-trace calculation.

Although not strictly a secondary keyword of **DIRECTION-SPACE**, the **DELTA0-AVE** secondary keyword is required whenever **DIRECTION-SPACE** is present.

The secondary keyword **RAYTRACE** causes an uncollided kerma calculation to be performed. This calculation performs a ray-tracing activity for every angle bin produced by the direction space procedure. The photon surface source is attenuated along the centroids of each angular bin using the total photon interaction cross section. The resulting photon source is folded with the photon kerma cross section for the specified detector material. The kerma is presented in the usual direction space output format. The RAYTRACE keyword is not allowed with the **USER** secondary keyword of **NBINT**. No other output is produced unless the additional secondary keyword **RAYPRINT** is used.

The **RAYPRINT** secondary keyword allows the user to print out the ray segment data for any or all of the rays generated with the RAYTRACE keyword. Ray segment data contains all the information needed to perform a 1-D transport calculation along the path of a ray. **RAYPRINT** with no parameters prints out ray segment data for all rays. If [parameter(1)] is present, it is the number of rays for which the user is requesting ray segment data and requires [parameter(1)] pairs of angles beginning on the next line of input. The angle pairs correspond to the theta (polar angle, 0 to 180 degrees) and phi (azimuthal angle, 0 to 360 degrees) of each angular bin for which the user wants ray segment data printed. The specified angles do not have to be exactly the angular bin centroids, ITS will find the angular bin that the given pair falls in and print the ray segment data.

46. **PHOTON-VOLUME-SOURCE** *(Adjoint Only)*

    See ELECTRON-VOLUME-SOURCE

47. **POSITION** *(Forward Only)*

    Syntax: POSITION

    Example: POSITION

    Default: The source is located at the origin.

    This keyword defines the position of the source. It must be followed by one of the following sub-keywords to further describe its spatial distribution:

    (a) **POINT**

    Syntax: POINT [parameter(1)] (TIGER codes)
    or POINT [parameter(1)] [parameter(2)] [parameter(3)] (non-TIGER codes)
    Example: POINT 2.0 (TIGER codes)
    or POINT 0.0 0.0 2.0 (non-TIGER codes)
    Default: The source is a point source at the origin.

    This sub-keyword specifies that the source is a point. For TIGER, the single location is the z coordinate of the point. For the non-TIGER codes, the three parameters specify the x, y and z coordinates, respectively.

    The default directional distribution is mono-directional in the positive z-direction, but the reference direction and source distribution may be specified by the DIRECTION keyword.

    (b) **LINE** *(CYLTRAN and ACCEPT Only)*

    Syntax: LINE

Example: LINE
3.2 8.1 5.5
9.1 3.4 3.0

Default: There is no default LINE distribution.

This keyword specifies the source as a line. The command line following the keyword contains the x, y, and z coordinates of one end of the line. The next command line contains the x, y, and z coordinates of the other end of the line. The source is sampled uniformly along the line segment.

The default directional distribution is mono-directional in the positive z-direction, but the reference direction and source distribution may be specified by the DIRECTION keyword.

(c) **DISK** *(CYLTRAN and ACCEPT Only)*

Syntax: DISK [parameter(1)] [parameter(2)] [parameter(3)]

Example: DISK 0.0 0.0 2.0

Default: There is no default DISK spatial distribution.

This keyword specifies the source as a disk. The three parameters specify the x, y, and z coordinates of the location of the center of the disk.

The default directional distribution is mono-directional in the direction of the orientation of the disk, but may be specified with the DIRECTION keyword.

This sub-keyword should be followed by the two following tertiary keywords:

i. **RADIUS**

Syntax: RADIUS [parameter(1)] [keyword]

Example: RADIUS 3.5 RADIAL-BIASING

Default: Zero radius point source.

[parameter(1)] specifies the radius of the disk source. This parameter may be followed by the keyword **RADIAL-BIASING** that will cause source particles to be sampled uniformly in radius (rather than uniformly in area).

ii. **ORIENTATION** *(Not yet functional)*

Syntax: ORIENTATION [parameter(1)] [parameter(2)]

Example: ORIENTATION

Default: The orientation of the disk is in the positive z-direction.

For now, the orientation of the disk is set by the reference direction under the DIRECTION keyword.

(d) **ANNULUS** *(CYLTRAN and ACCEPT Only)*

Syntax: ANNULUS [parameter(1)] [parameter(2)] [parameter(3)] [keyword] [parameter(4)]

Example 1: ANNULUS 0.0 0.0 2.0
4.0 2.0 RADIAL-BIASING

Example 2: ANNULUS 0.0 0.0 2.0 PROFILE 4
0.0 0.5 3.0 1.0
2.1 2.4 2.9 3.1

Default: There is no default ANNULUS spatial distribution. The orientation of the annulus is in the positive z-direction (or may be defined by the reference direction specified with the DIRECTION keyword).

This keyword specifies the source as an annulus. The first three parameters specify the x, y, and z coordinates of the location of the center of the annulus.

The default directional distribution is mono-directional in the direction of the orientation of the disk, but may be specified with the DIRECTION keyword.

If the optional PROFILE keyword is not present, then the following line should contain two parameters that specify the outer and inner radii of the annulus. These parameters may be followed by the keyword **RADIAL-BIASING** that will cause source particles to be sampled uniformly in radius.

If the **PROFILE** keyword is present, the user may specify a radial distribution to be sampled from. [parameter(4)] following the PROFILE keyword specifies the number of radial bins. Starting on the following line, [parameter(4)] values specify the cumulative probability array. This is followed by [parameter(4)] values specifying the corresponding radial array.

(e) **RECTANGLE** *(CYLTRAN and ACCEPT Only)*

Syntax: RECTANGLE

Example: RECTANGLE
1.0 1.0 0.0
1.0 2.0 0.0
3.0 2.0 0.0

Default: There is no default RECTANGLE spatial distribution.

This keyword specifies the source as a rectangle. The three lines following the keyword must contain the x, y, and z coordinates of vectors V1, V2, and V3 specifying 3 corners of the rectangle. (V1-V2) must be orthogonal to (V3-V2), and the reference direction is defined by (V1-V2) X (V3-V2).

(f) **SURFACE**

Syntax: SURFACE [keyword] (non-ACCEPT)
or SURFACE [parameter(1-3)] [keyword] (CYLTRAN)
or SURFACE [parameter(1)] BODY [parameter(2)] [keyword] (ACCEPT)

Example: SURFACE ZMAX (non-ACCEPT)
or SURFACE 1.0 2.5 5.0 OUTWARD (CYLTRAN)
or SURFACE 3 BODY 2 (ACCEPT)

Default: There is no default for the surface source option.

This sub-keyword specifies the surface index through which the source particles will be started. For TIGER and CYLTRAN, surface **ZMIN** refers to the minimum-z surface and surface **ZMAX** is the maximum-z surface. For CYLTRAN, surface **RMAX** is the lateral escape surface at maximum radius.

For CYLTRAN, a arbitrary cylindrical surface can be specified. [parameter(1)] is the lower Z coordinate. [parameter(2)] is the upper Z coordinate. [parameter(3)] is the radius of cylindrical surface. By default the reference direction is inward normal, but the reference direction may be defined as outward normal by using the keyword **OUTWARD**. The reference direction cannot be changed with the DIRECTION keyword.

For ACCEPT, it is necessary to specify both the surface index number and the associated **BODY** number. Figure 2 illustrates how many of the body types used in ACCEPT have their surfaces numbered. In the case of the arbitrary polyhedron (ARB), the user explicitly specifies the order in the input. The right circular cylinder is numbered the same as the truncated right-circular cone

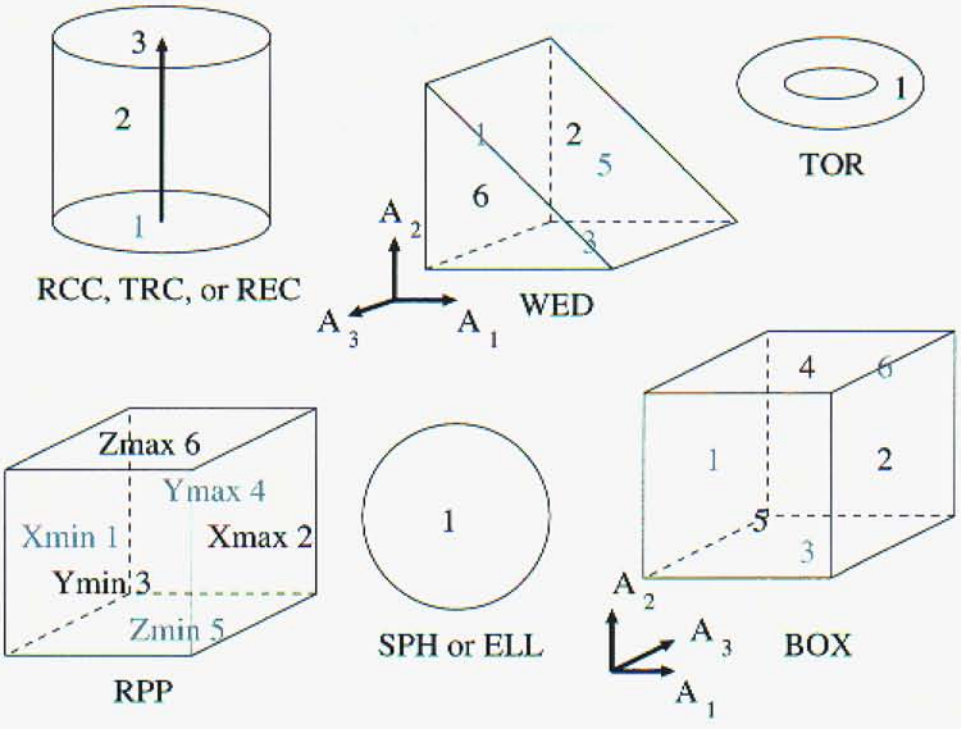


Figure 2. Surface indices for ACCEPT bodies

(TRC). The rectangular parallelepiped (RPP) is numbered the same as the BOX where A1 points in the positive x-direction, A2 points in the positive y-direction, and A3 points in the positive z-direction. The user need not be concerned if the desired surface has a unique description for the way the user has specified the geometry. By default the reference direction is inward normal, but the reference direction may be defined as outward normal by using the keyword **OUTWARD**. The reference direction cannot be changed with the DIRECTION keyword.

(g) **UNIFORM-ISOTROPIC-FLUX** *(CYLTRAN and ACCEPT Only)*

Syntax: UNIFORM-ISOTROPIC-FLUX [parameter(1)]

Example: UNIFORM-ISOTROPIC-FLUX [parameter(1)]

Default: There is no default UNIFORM-ISOTROPIC-FLUX source.

This keyword allows for the simulation of a uniform isotropic radiation field. The reference direction is inward normal, and the distribution is cosine-law on the surface. The reference direction cannot be changed with the DIRECTION keyword.

For CYLTRAN, no parameter is necessary; a cylinder sufficient to surround the problem is automatically used.

For ACCEPT, [parameter(1)] specifies the index of the body over which the source will be sampled. **WARNING:** There is no diagnostic to insure that anywhere outside of this body is the escape zone.

(h) **VOLUME**

Syntax: VOLUME [parameter(1)] [keyword] [parameter(2)]

Example: VOLUME 3

Example: VOLUME 3 SHELL 1.5 (ACCEPT codes)

Default: There is no default for the volume source option.

For TIGER, [parameter(1)] is the input-zone index.

For ACCEPT, [parameter(1)] is a body index. For a SPH or RCC body, the keyword **SHELL** may be used with [parameter(2)] specifying the inner radius of the spherical or cylindrical shell. For ACCEPT, the body referred to need not be part of the actual zone description of the geometry.

48. **PRINT-ALL**

Syntax: PRINT-ALL

Example: PRINT-ALL

Default: Only the cumulative results for the final batch will be written to the output file.

This primary keyword causes the cumulative results from all batches to be written to the output file (Fortran unit 6).

49. **PULSE-HEIGHT** *(ITS Only)*

Syntax: PULSE-HEIGHT [parameter(1)] [parameter(2)]

Example: PULSE-HEIGHT 4 7

Default: No spectrum of absorbed energy will be calculated.

This keyword causes the spectrum of absorbed energy to be calculated for input zones [parameter(1)] through [parameter(2)]. These parameters correspond to the order of the input zones as those zones were input. If the parameters are left blank, the spectrum of absorbed energy will be calculated for the entire geometry. Certain biasing schemes, such as those activated by the sub-keywords SCALE-BREMS and SCALE-IMPACT, are inconsistent with this calculation; PULSE-HEIGHT will cause them to be deactivated (a message so informing the user is written to the output file). The following secondary keyword describes the energy bin structure used in tallying the spectrum of absorbed energy.

(a) **NBINE**

Syntax: NBINE [parameter(1)] [keyword]

Example: NBINE 6 USER
1.99999 1.0 0.5 0.25 0.00001 0.0

Default: ten bins of equal width plus total absorption and escape (i.e., 12 bins total).

If [keyword] is not specified then [parameter(1)] is the number of desired equal-width bins plus 2 (to account for both total absorption and escape). If [keyword] is USER, [parameter (1)] is the number of bin energies to be read. The only choice for [keyword] is:

i. **USER** - User defined energy grid. The code will then read the lower bounds of the energy bins (MeV) in descending order as in the above example. The maximum lower bound must be less than the maximum source energy. In the above example, the first lower bound and the last two lower bounds were chosen to insure that total absorption (full source particle energy absorbed in the selected region) and total escape (no energy absorbed in selected region for a given source particle), respectively, would be accounted for.

Note that the primary keyword alone, with no other parameters or keywords, will result in the calculation of the spectrum of absorbed energy for the entire geometry using the default bin structure.

50. **RANDOM-NUMBER**

Syntax: RANDOM-NUMBER [parameter(1)]

Example: RANDOM-NUMBER

4265641542

Default: 0 (converted to 5^{19})

[parameter(1)] is the initial random number seed for the Monte Carlo run. This keyword can be used to start a run with the final random number from an earlier run for which a dump file does not exist.

For RNG1, this keyword can also be used in debugging to isolate the offending primary history. For a similar purpose, the more sophisticated user can use this keyword in conjunction with a print of the initial random number seed of a source particle, IRSAV.

For RNG2 and RNG3, the state of the RNG is more than a single seed. See the RESTART-HISTORY keyword for a method of debugging with those generators.

See the Random Number Generators section for further discussion of issues concerning random number routines.

51. **REFLECTION-ZONE** *(ACCEPT and ACCEPTP Only)*

Syntax: REFLECTION-ZONE [parameter(1)]

Example: REFLECTION-ZONE 20

Default: No reflection zone.

Particles undergo specular reflection at the boundaries of the selected zone. Note: Source particles cannot be initiated in the reflection zone, and no particles can enter the reflection zone. This keyword cannot be used with MCODES.

52. **RESTART**

Syntax: RESTART

Example: RESTART

Default: A new calculation is performed instead of a restart.

Problem is restarted at the batch number of the data in the "fort.14" file. A dump file must have been written, saved, and named "fort.14" (see FILE-NAMES keyword). The batch size on the restart run must be the same as those on the dump file to permit accurate computation of statistical uncertainties. If specified otherwise, the batch size will be set equal to the batch size used to generate the dump file. The total number of batches and histories specified in the restart input file should be the desired additional batches and histories.

53. **RESTART-HISTORY** *(RNG2 and RNG3 Only)*

Syntax: RESTART-HISTORY

Example: RESTART-HISTORY

Default: A new calculation is performed instead of a restart.

This keyword specifies that a restart will be executed using the state of the random number generator specified in the file "rngstate.dump". This file is written when the program exits due to an error in execution. It contains the state of the random number generator at the start of the particle history in which the error occurred. Only the offending history will be executed. This allows the user to repeat the single history in which an error occurs.

In parallel, the restart must be performed with STATIC load balancing, since the RNG state is not passed to subtasks.

54. **SIMPLE-BREMS** *(ITS Only)*

Syntax: SIMPLE-BREMS

Example: SIMPLE-BREMS

Default: more accurate bremsstrahlung distributions.

This keyword specifies that bremsstrahlung distributions are to be used corresponding to ITS version 2.1.

55. **SOURCE-SURFACES** *(MITS Adjoint Only)*

Syntax: SOURCE-SURFACES [parameter(1)]

Example: SOURCE-SURFACES 3

Default: All surfaces of the escape zone (tallied as a single surface) for ACCEPT. Both surfaces (ZMIN and ZMAX) for TIGER.

This specifies the number of surfaces on which either ELECTRON-SURFACE-SOURCEs, PHOTON-SURFACE-SOURCEs, or NEUTRON-SURFACE-SOURCEs will be displayed. These specified surfaces should correspond to adjuncton-escape surfaces. This keyword must be followed by [parameter(1)] separate lines of the secondary keyword SURFACE.

Any specification for CAD will be ignored. All escaping particles are scored in a single body/surface tally.

(a) **SURFACE**

Syntax: SURFACE [keyword] (non-ACCEPT codes)
or SURFACE [parameter(1)] BODY [parameter(2)] (ACCEPT codes)
Example: SURFACE ZMAX (non-ACCEPT codes)
or SURFACE 3 BODY 2 (ACCEPT codes)

This sub-keyword specifies the surface index through which the escaping particles will be calculated.

For TIGER and CYLTRAN, surface **ZMIN** refers to the minimum-z surface and surface **ZMAX** is the maximum-z surface. For CYLTRAN, surface **RMAX** is the lateral escape surface at maximum radius. For ACCEPT, it is necessary to specify both the surface index and the **BODY** number. Figure 2illustrates how many of the body types used in ACCEPT have their surfaces numbered. In the case of the arbitrary polyhedron (ARB), the user explicitly specifies the order in the input. The right circular cylinder is numbered the same as the truncated right-circular cone

(TRC). The rectangular parallelepiped (RPP) is numbered the same as the BOX where A1 points in the positive x-direction, A2 points in the positive y-direction, and A3 points in the positive z-direction.

56. **SPECTRUM**

Syntax: SPECTRUM [parameter(1)] [keyword] [keyword]

Example: SPECTRUM 5

1.00 0.80 0.76 0.53 0.00

5.0 4.0 3.0 2.5 2.0

Example: SPECTRUM 5 NUMBER-PER-BIN

0.40 0.08 0.46 1.06

5.0 4.0 3.0 2.5 2.0

Example: SPECTRUM 5 NUMBER-PER-BIN-PER-MEV

0.20 0.04 0.46 1.06

5.0 4.0 3.0 2.5 2.0

Default: mono-energetic source (ITS) or mono-group source (MITS)

In forward mode, this keyword specifies that the energy distribution is a spectrum. [parameter(1)] is the number of energy grid points describing the spectrum (or one more than the number of energy bins in the spectrum). The spectrum follows this keyword, decreasing monotonically to 0.0. The corresponding spectrum energy bin grid is given on the next line. If the **NUMBER-PER-BIN** or **NUMBER-PER-BIN-PER-MEV** keywords appear, the code will convert the distribution to a cumulative distribution internally. The examples given above result in identical spectra.

In any of these formats, the spectrum does not have to be normalized (e.g., the cumulative distribution of the spectrum does not have to begin with 1.0.) If the spectrum is not normalized, the code will produce a warning, normalize the spectrum, and proceed with the calculation. The distribution is normalized based only on the spectrum data provided. If a portion of the spectrum falls below the cutoff energy, source particles sampled from below the cutoff will not be tracked (but information about the fraction of such rejected particles will appear in the output).

The spectrum may contain line sources and ranges of the spectrum with zero probability, such as in the following example:

Example: SPECTRUM 4

1.0 0.5 0.5 0.0

1.3325 1.3325 1.1732 1.1732

The user may bias the energy sampling of source particles. If the **BIASED** keyword appears on the same line as the SPECTRUM keyword, the first spectrum read will be the spectrum sampled from and the second spectrum will be the true spectrum of the source particles. Both spectra must be on the same energy grid. In the following example, the 1.3325 MeV line will be sampled 3 times more often by the code than the 1.1732 MeV line (but the particle weights will be adjusted to account for the fact that the true source has equal probability of each line).

Example: SPECTRUM 4 BIASED

1.0 0.25 0.25 0.0
1.0 0.5 0.5 0.0
1.3325 1.3325 1.1732 1.1732

In adjoint mode, this keyword allows the detector response to be calculated by folding with multiple forward source spectra during a single calculation.

Syntax: SPECTRUM [parameter(1)] ADJOINT-SPECTRA [keyword]
Example: SPECTRUM 2 ADJOINT-SPECTRA PHOTON
5 2.31
4 1.56
1.00 0.80 0.76 0.53 0.00
14.2 13.1 10.1 8.6 5.4
1.00 0.50 0.25 0.0
14.0 12.0 9.0 5.0
Example: SPECTRUM 1 ADJOINT-SPECTRA PHOTON NUMBER-PER-BIN
5
0.462 0.0924 0.5313 1.2243
14.2 13.1 10.1 8.6 5.4
Default: The detector response is only calculated by folding with a flat forward spectrum.

In adjoint mode, the secondary keyword **ADJOINT-SPECTRA** must be included on the same line as the SPECTRUM keyword, as well as a keyword specifying the type of source particle as one of **PHOTON, ELECTRON**, or **NEUTRON**. [parameter(1)] specifies the number of forward spectra. [parameter(1)] lines must follow, each containing two parameters: the first parameter specifies the number of energy grid values (the number of bins plus one) in the corresponding spectrum, and the second parameter specifies the magnitude (or source strength) of the spectrum. Then, 2×[parameter(1)] lists must follow. For each spectrum, the first list is the distribution, and the second list is the corresponding energy bin grid. The energies need not correspond to the energy divisions on the cross section set, however source energies cannot include energies for which cross sections are not available. The keywords **NUMBER-PER-BIN** or **NUMBER-PER-BIN-PER-MEV** can be added to the keyword line to specify those formats for the spectra, otherwise the format is assumed to be a cumulative distribution of the number of particles per bin.

WARNING: The results are always multiplied by the source strength factor. It is also possible to specify an unnormalized spectrum. While it is possible to apply a source strength factor to an unnormalized spectrum, this is not likely to be a desired feature. If no source strength factor is specified, none will be applied (that is, the factor will be set to 1.0).

The default is scoring with a flat forward spectrum and is included in the output if this keyword is not used. If this keyword is used to specify forward spectra, output will be generated based on folding with the forward spectra. In either case results will be given in the energy bin structure specified with the **NBINE** sub-keyword of the SURFACE-SOURCE selected. However, in the case of the default, results will correspond with folding against a flat spectrum of unit strength for each energy span reported. That is, the unit strength is applied to each energy span, not to the entire energy span of the problem. If this keyword is used, results will correspond to the response due to source particles in the energy span reported.

57. **TASKS** *(MPI Only)*

Syntax: TASKS [parameter(1)] [parameter(2)] [parameter(3)]

Example: TASKS 50 100 1.1

Default: Number of processors available is used. Intermediary output after every batch. No termination due to stray processes.

This keyword applies only for parallel processing. [parameter(1)] specifies the number of processors to which batches can be distributed. [parameter(2)] specifies the number of batches between intermediary outputs. [parameter(3)] is a factor multiplying the running average of the batch time such that if any batch time exceeds this time, the batch is considered to be in an infinite loop and the run will be terminated.

If the run is terminated for an assumed infinite loop, the random number seed for that batch is output so that, subsequently, that batch alone can be run by using that random number seed as the parameter for the RANDOM-NUMBER keyword. If the code itself detects an error condition that is not an infinite loop and calls ABORTX, the number of random numbers to the beginning of the offending history is output and the "rngstate.dump" file is written. The user can subsequently run only the offending batch (with the RANDOM-NUMBER keyword) or the offending history (with the RESTART-HISTORY keyword).

58. **TITLE**

Syntax: TITLE [parameter(1)]

Example: TITLE

Adjoint Dose calculation in Al box in Satellite GPS-4

Default: no title

This keyword signals that the next line of input contains [parameter(1)], which is a title of up to 80 columns that will be written to the output file and will be used as the title on any plots that are generated.

Last Modified: January 9, 2004

9 TIGER Geometry

The geometry of the TIGER codes is the simplest of the ITS member codes. It is strictly one dimensional. A particle trajectory is described only in terms of the z coordinate of position and the z direction cosine. Nevertheless, this is often all that is necessary, and, because the TIGER codes are the fastest and simplest to use, they should always be considered. They are especially useful in obtaining accurate answers to questions involving very basic transport phenomena.

9.1 Problem Geometry

Beginning at z=0.0, layers are stacked along the positive z axis according to the order in which they are read in as described under keyword GEOMETRY. For each layer the user must define: (a) the material index, (b) the number of subzones into which the layer is to be divided for purposes of scoring charge deposition, energy deposition, and particle flux (see the discussion of Automatic Subzoning), and (c) the thickness of the layer. The material indices are defined by the order in which the materials are specified in executing the cross-section generating code.

Interior voids must not be defined. Voids within the geometry have no effect upon one-dimensional transport. Omission of voids allows for increased efficiency in the calculation and is a requirement for TIGER geometry descriptions.

9.2 Conventions for Escaping Particles

In addition to quantities internal to the problem geometry such as charge deposition, energy deposition and particle flux, radiation that escapes may also be scored. Because geometry is defined as infinite slabs with only a finite z-dimension, particle escape is classified as one of:

1. Radiation that escapes from the maximum-z boundary of the problem.
2. Radiation that escapes from the minimum-z boundary of the problem.

In forward mode, these definitions may be applied by the user with the ESCAPE-SURFACES keyword to specify where escaping particles are to be tallied. In adjoint mode, the SOURCE-SURFACES keyword to specify where escaping adjuncton particles are to be tallied. The default in either mode is to perform tallies on both surfaces.

Last Modified: January 9, 2004

10 CYLTRAN Geometry

There are a variety of experimental problems in which the symmetry requirements of the CYLTRAN codes are satisfied to a good approximation. This is especially true in those experiments for which the radiation source is itself a cylindrical beam, as in the case of many pulsed and steady-state electron accelerators. The only essential requirement, however, is that the material geometry, as specified by the input zones, be cylindrically symmetric. The trajectories themselves are fully three dimensional. In a code modification, the more sophisticated user may wish to define a non-axisymmetric source or, in the case of CYLTRANM, a non-axisymmetric field configuration, along with whatever azimuthal tallies he desires. Note that the logic is already included for scoring azimuthally-dependent escape distributions (see keywords ELECTRON-ESCAPE, PHOTON-ESCAPE, and NEUTRON-ESCAPE) and azimuthally-dependent charge deposition, energy deposition, and particle fluxes (see keywords ELECTRON-FLUX, PHOTON-FLUX, NEUTRON-FLUX, and GEOMETRY, as well as the discussion of Automatic Subzoning).

10.1 Problem Geometry

The material geometry for the CYLTRAN codes consists of a right circular cylinder of finite length, the axis of which coincides with the z axis of the Cartesian system that describes the particle trajectories. The location of this cylinder, hereafter referred to as the problem cylinder, along the z axis is completely arbitrary. The entire volume within the problem cylinder must be specified in terms of material or void input zones, each of which is bounded by two and only two cylinders coaxial with the z axis and two and only two planes perpendicular to the z axis.

The material configuration is then conveniently described by the half section of the problem cylinder obtained by passing a plane through its axis. An example of such a half section is shown in Fig. 3. The horizontal base line is the axis of the problem cylinder, and the other horizontal lines are labeled by the radii of the corresponding cylindrical boundaries. The vertical lines are labeled by the z coordinate of the corresponding plane boundaries. The solid lines are actual material boundaries; the broken lines are not. The dashed lines are employed either to complete the perimeter of the problem cylinder half section or to break more complex zones of a given material (e.g., those having L-shaped half sections in Fig. 3) into the simpler input zones required by the code (i.e., zones whose half sections are rectangles). The dotted lines describe subzoning of a given input zone for purposes of obtaining charge deposition, energy deposition, and flux profiles.

Each zone in Fig. 3 is bounded by solid and/or dashed lines and contains a material index (circled). A zero index defines a void zone; otherwise, the material indices are defined by the order in which the materials are specified in executing the cross-section generating code. Each of these input zones requires a single input card for its description. The dotted lines illustrate subzoning of a particular input zone into equal axial and/or radial increments. Azimuthal subzoning is also possible as discussed under the GEOMETRY keyword. It also follows from that discussion that separate input cards describing these subzones are not required. This description is accomplished internally by the code using the subzoning parameters specified on the input card describing the input zone. This feature allows the user to obtain three-dimensional charge deposition, energy deposition, and flux profiles within a given input zone with a single input card.

The following input cards describe the problem geometry illustrated in Fig. 3.

```
GEOMETRY 6
-2.50 -2.00 0.00 1.25 1
-1.50 0.00 0.00 1.50 2 1 3 2
```

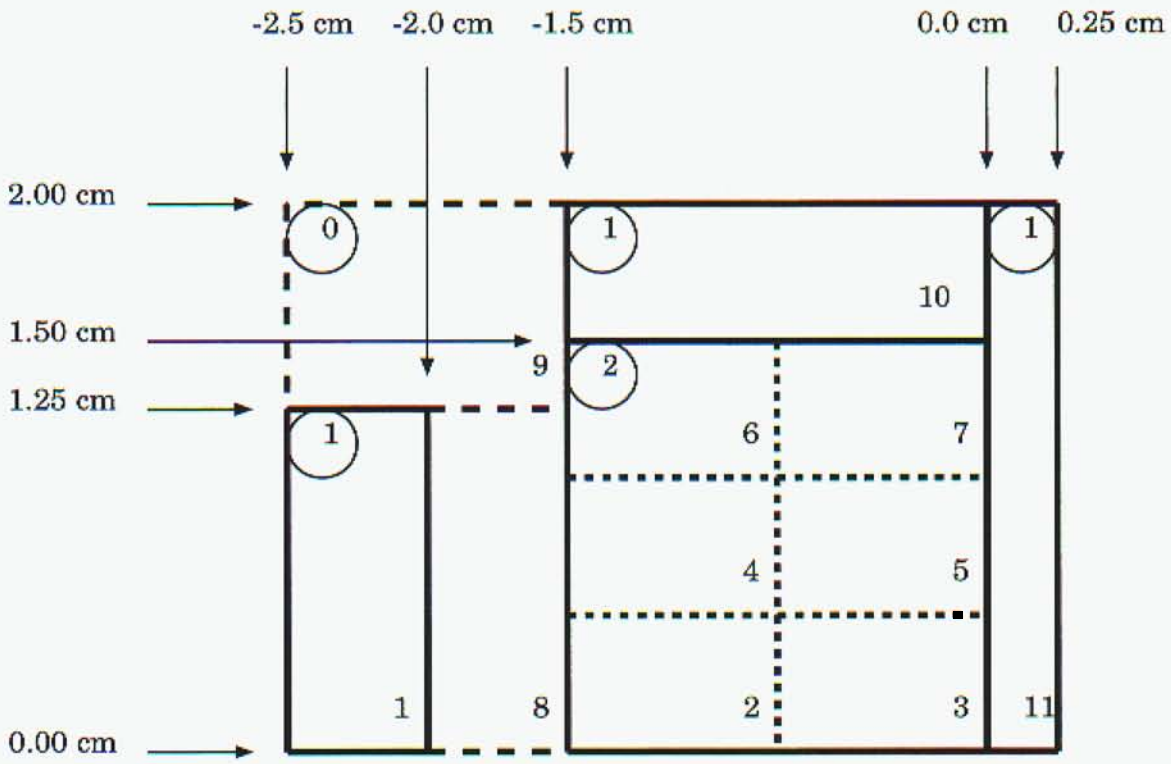


Figure 3. Example of the half section of a problem cylinder

-2.00 -1.50 0.00 1.25 0
-2.50 -1.50 1.25 2.00 0
-1.50 0.00 1.50 2.00 1
0.00 0.25 0.00 2.00 1

The numbers in the lower right hand corners of the subzones demonstrate how the code internally numbers these subzones. Subzone numbers are immediately assigned as each geometry card is read. Therefore, subzones are numbered before the next card is read. In the example, the second card, which describes the second input zone, generates 6 subzones (subzones 2-7). The next input zone, therefore, has a subzone number of 8. It is important that the user understand this numbering scheme in order to properly interpret spatially-dependent outputs.

In the case of CYLTRANM, the user may specify the presence of macroscopic electric fields (in voids only) and/or magnetic fields with an additional parameter in the zone description. Specification of this parameter is described under the GEOMETRY keyword.

10.2 Conventions for Escaping Particles

In addition to quantities internal to the problem cylinder, such as charge deposition, energy deposition, and particle flux, radiation that escapes from the problem cylinder may also be scored. Because the geometry is defined in cylindrical coordinates of thickness z and radius r, particle escape is classified according as one of:

1. Radiation that escapes from the maximum-z boundary of the problem cylinder.
2. Radiation that escapes from the minimum-z boundary of the problem cylinder.
3. Radiation that escapes from the maximum-r curved lateral boundary of the problem cylinder.

In forward mode, these definitions must be applied by the user with the ESCAPE-SURFACES keyword to specify where escaping particles are to be tallied. In adjoint mode, the SOURCE-SURFACES keyword to specify where escaping adjuncton particles are to be tallied. The default in either mode is to perform tallies at all three surfaces.

Last Modified: February 16, 2004

11 ACCEPT Geometry

The ACCEPT codes provide experimenters and theorists with a method for the routine solution of coupled electron/photon transport through three-dimensional multimaterial geometries described by the combinatorial method. In the combinatorial scheme, the problem input zones are built up out of primitive bodies. This is in contrast to more traditional schemes that define the zones in terms of bounding surfaces. The SANDYL code is an example of the latter in that it makes use of a system of paraxial quadratic surfaces and Cartesian planes in order to define the problem zones. We find the combinatorial method of specifying input zones in terms of solid bodies to be simpler, more intuitive, and less ambiguous than specification in terms of boundary surfaces. The combinatorial scheme also learns as the calculation progresses; at any particular time it makes use of information obtained from past experience in order to improve the efficiency of its search procedures used in particle tracking. This same learning ability precludes the requirement, typical of many other geometry schemes, for inputting a substantial amount of tracking information.

11.1 Problem Geometry

With the ACCEPT codes the user employs the combinatorial-geometry method in order to describe the three-dimensional material configuration of the problem. This task is accomplished in five distinct steps:

1. Define the location and orientation of each solid geometrical body required for specifying the input zones.
2. Specify the input zones as combinations of these bodies.
3. Specify zones to be subzoned and subzoning schemes, if necessary.
4. Specify the volumes of the subzones, if necessary.
5. Specify the material in each input zone.

11.1.1 Body Definition

The combinatorial-geometry method requires a library of geometrical body types from which the user may choose in order to describe his problem configuration. The information required to specify each body type in a three-dimensional Cartesian system is as follows:

1. Rectangular Parallelepiped (RPP) – Specify the minimum and maximum values of the x, y and z coordinates that bound a rectangular parallelepiped whose six sides are perpendicular to the coordinate axes.
2. Sphere (SPH) – Specify the components of the radius vector **V** to the center of the sphere and the radius R of the sphere.
3. Right Circular Cylinder (RCC) – Specify the components of a radius vector **V** to the center of one base, the components of a vector **H** from the center of that base to the center of the other base, and the radius R of the cylinder.

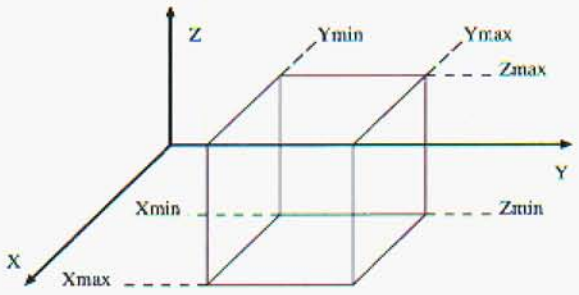


Figure 4. Rectangular Parallelepiped (RPP)

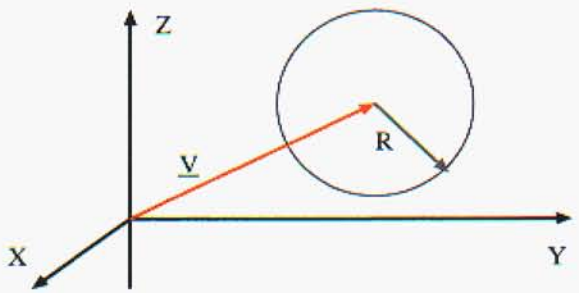


Figure 5. Sphere (SPH)

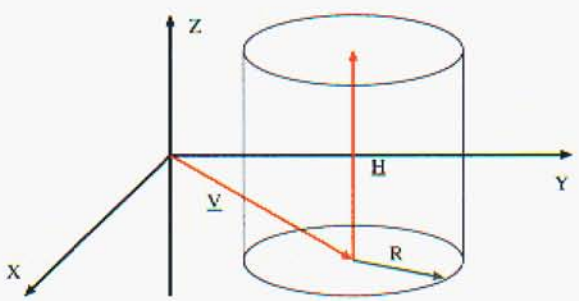


Figure 6. Right Circular Cylinder (RCC)

4. Right Elliptical Cylinder (REC) – Specify the components of a radius vector **V** to the center of one of the elliptical bases, the components of a vector **H** from the center of that base to the center of the other base, and the components of two vectors $\mathbf{R}_1$ and $\mathbf{R}_2$ that define the major and minor axes, respectively, of the bases.

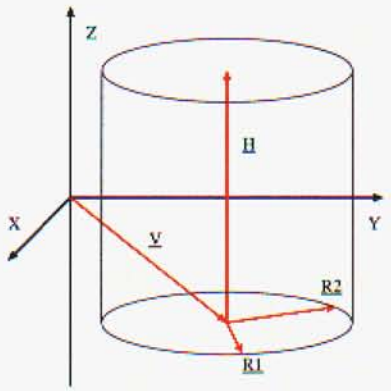


Figure 7. Right Elliptical Cylinder (REC)

5. Truncated Right-Angle Cone (TRC) – Specify the components of a radius vector **V** to the center of one base, the components of a vector **H** from the center of that base to the center of the other base, and the radii R_1 and R_2 of the first and second bases, respectively.

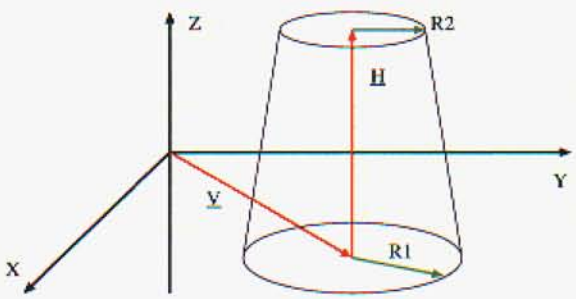


Figure 8. Truncated Right-Angle Cone (TRC)

6. Ellipsoid (ELL) – Specify the components of the radius vectors $\mathbf{V}_1$ and $\mathbf{V}_2$ to the foci of the prolate ellipsoid and the length of the major axis R. ITS can not accept an oblate ellipsoid. The foci of a prolate ellipsoid are on the major axis, the axis about which the ellipse is rotated to form the body. The center of the body lies halfway between the foci. The square of the length of the major axis equals the sum of the square of the length of the minor axis plus the square of the distance between the foci.

7. Wedge (WED) – Specify the components of a radius vector **V** to one of the corners and the components of three mutually perpendicular vectors $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$ starting at that corner and defining the wedge such that $\mathbf{a}_1$ and $\mathbf{a}_2$ are the two legs of the right triangle of the wedge.

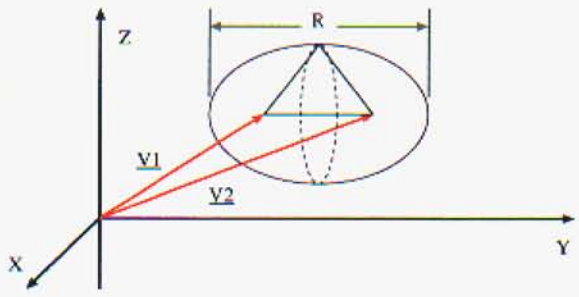


Figure 9. Ellipsoid (ELL)

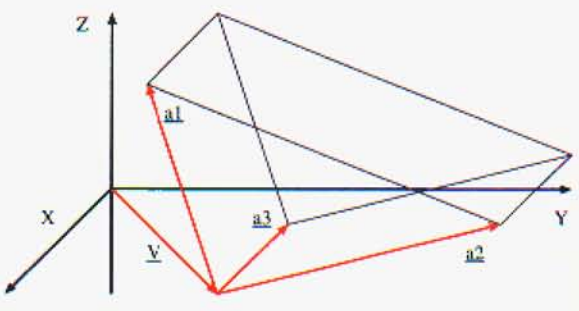


Figure 10. Right Angle Wedge (WED)

8. Box (BOX) – Specify the components of a radius vector **V** to one of the corners and the components of three mutually perpendicular vectors $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$ starting at that corner and defining a rectangular parallelepiped of arbitrary orientation.

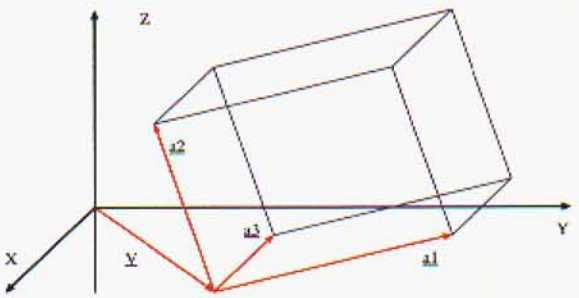


Figure 11. Box (BOX)

9. Arbitrary Polyhedron (ARB) – Specify the components of k (k = 4, 5, 6, 7, or 8) radius vectors, $\mathbf{V}_1$ through $\mathbf{V}_k$, to the corners of an arbitrary non-reentrant polyhedron of up to six sides, and specify the indices of the corners of each face by means of a series of four-digit numbers between "1230" and "8765" (enter zero for the fourth index of a three-cornered face). The digits must appear in either clockwise or counterclockwise order.

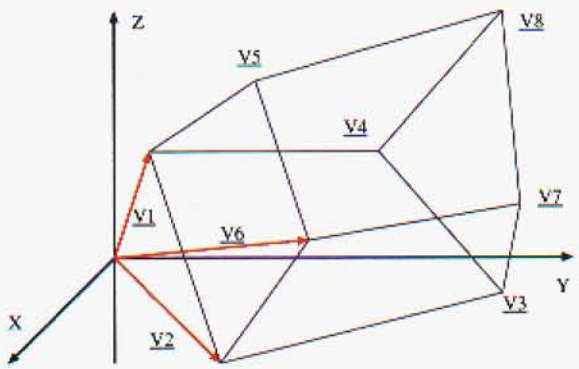


Figure 12. Arbitrary Polyhedron (ARB)

10. Torus (TOR) – The vector **V** specifies the coordinates of the centroid of the torus, the unit vector **H** specifies the axis of revolution, the major radius R specifies the distance from the centroid of the torus to the center of the ellipse to be rotated, the radius R_H specifies the axis of the ellipse parallel to the **H** vector, and the radius R_p specifies the other axis of the ellipse. For now, only circular ($R_H = R_p$) tori are allowed.

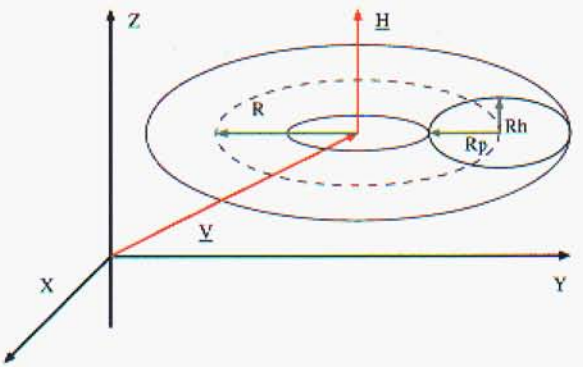


Figure 13. Torus (TOR)

11.1.2 Specification of Input Zones

Having defined the necessary geometrical bodies, the user must then resolve the entire problem geometry into input zones satisfying the following criteria:

1. An input zone may consist only of either a single homogeneous material or a void.
2. Every point of the problem geometry must lie within one and only one input zone.
3. The final input zone must be a void zone surrounding the rest of the problem geometry that is entered through a non-reentrant surface; any particle entering this zone is treated as an escape particle.

Input zones are specified as appropriate combinations of the previously defined bodies. Such combinations may be as simple as just a single body, or they may consist of complex intersections, unions and differences of various bodies. We illustrate the principles of input zone specification with the following examples where, for simplicity, we omit the escape zone. Each example involves only two zones, A and B, defined by the cross hatching in Fig. 14.

In Fig. 14a, zone A consists of a sphere, body #1, that is tangent to zone B, which consists of a right circular cylinder, body #2. Input zone specification is simply

A= +1 ,
B = +2 .

That is, **input zone A consists of all spatial points that lie within body #1**, and similarly for zone B.

In Fig. 14b, the sphere is inserted into a hole that has been cut in the cylinder so that

A = +1 ,
B = +2 -1 .

Thus, **input zone B consists of all spatial points that lie within body #2 AND not within body #1**. Input zone B is specified as the difference between two bodies.

In Fig. 14c, bodies #1 and #2 consist of the same homogeneous material (or void), but they are imbedded within a second right circular cylinder, body #3, of another material. The specification is

A = +1 OR +2 ,
B = +3 -1 -2 .

Thus, **input zone A consists of all spatial points that lie within EITHER body #1 OR body #2**. This is an example of input zone specification as a union of bodies.

In Fig. 14d, the intersection of body #1 and body #2 consists of a single homogeneous material; the rest of the space within body #3 is filled with another material. The specification is

A = +1 +2 ,
B = +3 -1 OR +3 -2 .

Thus, **input zone A consists of all spatial points that lie within body #1 AND within body #2.**

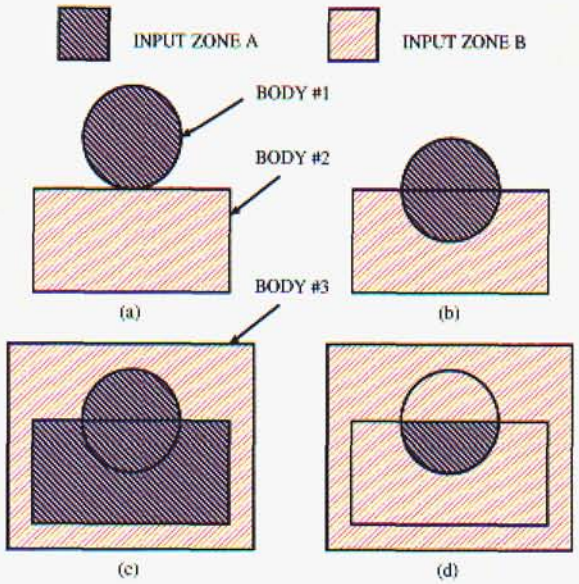


Figure 14. Illustration of various methods of combining bodies for specification of input zones

Note that:

1. The OR operator refers to all following body numbers until the next OR operator is reached or a new input zone is initiated.

2. The AND operator is implied before every body number that is not preceded by an explicit OR operator, except that the first OR operator of a union is an implied EITHER.

Though Figs. 14a and 14b are useful for demonstrating how input zones are constructed, they are not good examples of transport geometries because they are reentrant. By reentrant we mean that there are some paths by which escaping particles can reenter those geometries. They can be made non-reentrant by enclosing them completely in a non-reentrant body such as a sphere and letting the escape zone be the region outside the sphere.

11.1.3 Subzone Specification

In Version 2.0 we began implementing automatic subzoning features into the ACCEPT codes. (See the section on Automatic Subzoning for further details.) In addition to reductions in memory requirements and run time, this powerful option eliminates the burdensome task of otherwise generating an input-zone description for each individual subzone. The ACCEPT codes now feature the full three-dimensional subzoning of input zones consisting of a single body of type RCC, RPP, BOX, SPH, WED, TRC, and TOR and

subzoning for some multi-body input zones. Automatic subzoning is available for CAD zones based on RPP subzoning of the CAD bounding box. All of the available subzoning schemes can be used as non-conformal subzone overlays for obtaining simple profiles within complicated CG or CAD zones. Each subzoning scheme divides the subzone entity into equal intervals in three different dimensions based on three integers supplied by the user.

Single Body Subzoning The simplest type of subzoning involves a zone composed of a single body. The following is a description of how the three subzoning integers are used for these types:

1. RPP – The three integers correspond to subzoning along the three Cartesian directions, x, y, and z, respectively. Here, the body-based coordinate directions are the same as those of the laboratory system. Distances are measured along the axes from the point (Xmin, Ymin, Zmin) defined in Fig. 4.

2. BOX – The three integers correspond to subzoning along the three Cartesian directions, $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$, respectively, as defined in Fig. 11. Distances are measured from the point defined by the radius vector **V**.

3. WED – The three integers correspond to subzoning along the three Cartesian directions, $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_3$, respectively, as defined in Fig. 10. This subzoning is similar to the BOX, except that subzones are cut by the sloped surface of the wedge. Distances are measured from the point defined by the radius vector **V**.

4. RCC – The three integers correspond to subzoning azimuthally about the cylinder axis, in distance from the axis (radially), and in distance along the cylinder axis (axially) from the center of the base defined by the radius vector **V** in Fig. 6, respectively.

5. TRC – The three integers correspond to the same subzoning definitions used for the RCC: azimuthal, radial, and axial.

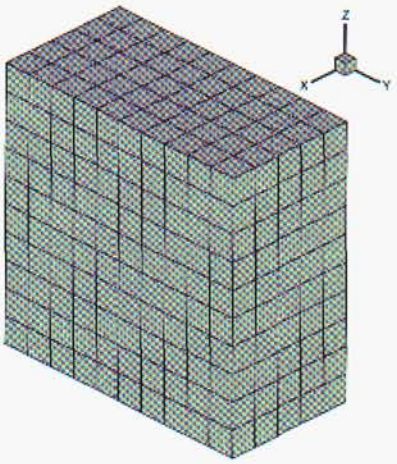
6. SPH – The three integers correspond to subzoning azimuthally about the laboratory z axis as measured with respect to the positive x axis, in polar angle as measured with respect to the laboratory z axis, and in distance from the center of the sphere (radially), respectively.

7. TOR – The three integers correspond to subzoning in the poloidal angle, the cross section radius of the torus, and the toroidal angle (angle about the axis of revolution). This is analogous to the subzoning of an RCC, where the torus is a cylinder with its axis wrapped in a circle. Thus, the RCC azimuthal angle is the TOR poloidal angle, the RCC radial coordinate is the cross section radius, and the RCC axial coordinate is the toroidal angle. Currently, subzoning is restricted to circular tori.

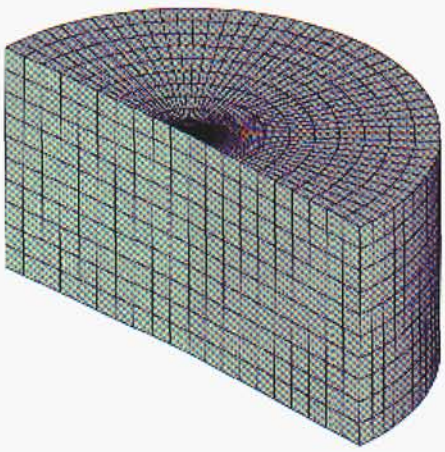
   The poloidal angle is the angle between the cross section radius vector **r** (currently, $\mathbf{r} = \mathbf{R}_H = \mathbf{R}_p$) and the major radius vector **R** in Fig. 13. The sense of rotation for the poloidal angle is such that a radius vector **r** in the direction of the unit vector **H** is a 90-degree poloidal angle. The poloidal subzones are created by dividing the 360-degree angle-space by the number of poloidal subzones requested.

   The subzoning of the cross section radius is currently limited to circular shells since the torus must be circular. The radial subzones are created by dividing the cross section radius equally into the number of cross section subzones requested.

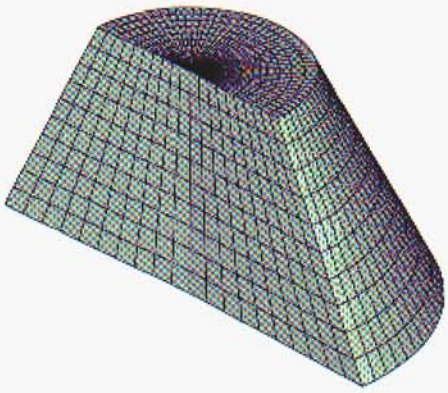
   The toroidal angle about the axis of revolution proceeds as the calculation of any azimuthal angle about an axis. In this case, the azimuthal angle is determined by the dot product of the perpendicular component of the radial vector **R** with the reference vector. The toroidal subzones are created by dividing the 360-degree angle-space by the number of toroidal subzones requested.



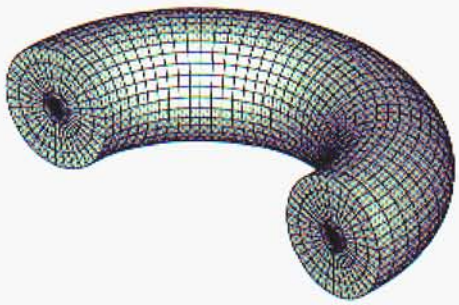
RPP subzoning



RCC subzoning



TRC subzoning



TOR subzoning

Multi-Body Subzoning There are currently 8 multi-body subzone entities available in ITS. These subzone entities can be divided into 2 types: closed shells and cylindrical-like shells. The closed shells are the SPH-SPH and TOR-TOR. Both of these require that the two bodies be concentric. The TOR-TOR must be composed of tori that share an axis of rotation, as shown in Fig. 15. Subzoning is based on the same coordinate systems as the single body subzoning. The number of "radial" subzones specifies the number of subzone layers between the inner and outer boundary of the shell.

The other 6 multi-body subzone entities all use cylindrical-like body-based coordinates for subzoning: azimuthal, radial, and axial. The azimuthal and axial boundaries have their usual meaning. The radial subzone boundaries are equally spaced between the inner and outer zone boundaries for all axial coordinates. This is best illustrated in the curved subzone boundaries of the SPH-RCC. For the SPH-RCC, the center of the sphere must lie on the axis of the RCC. In all other cases the bodies defining the inner and outer radii must be coaxial with each other. For the TRC-TRC, the wide parts of the frusta need not be on the same side.

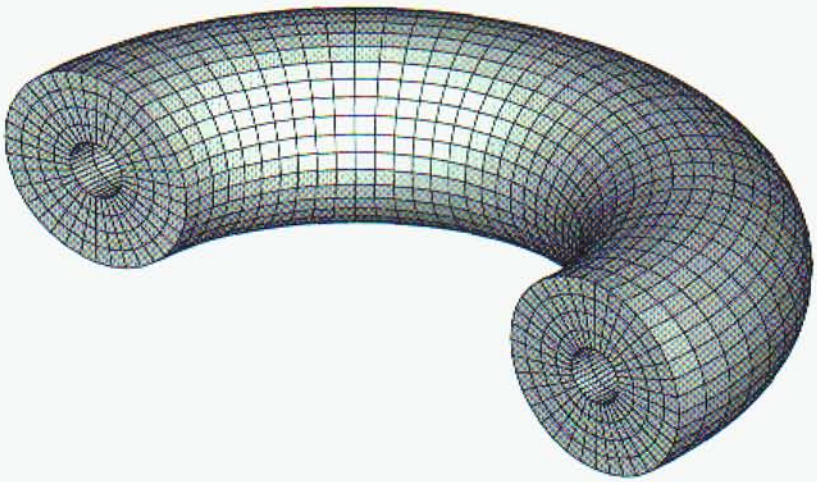
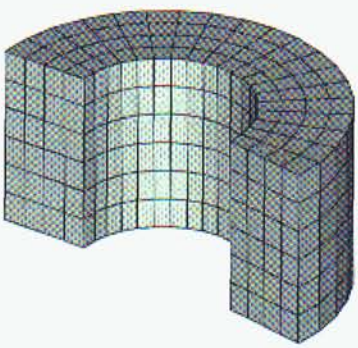
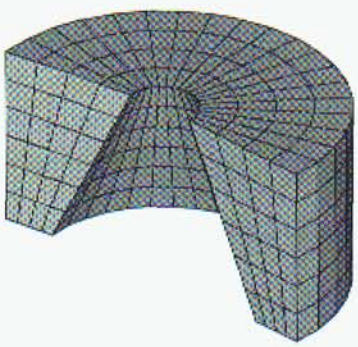


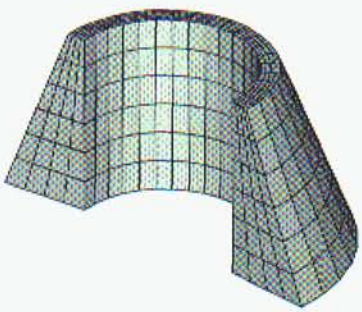
Figure 15. TOR-TOR subzoning



RCC-RCC subzoning

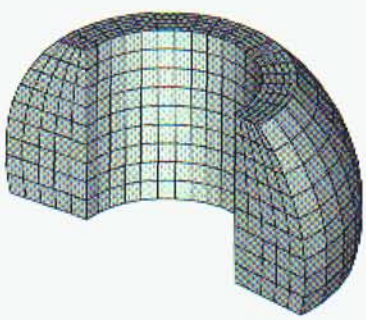


RCC-TRC subzoning

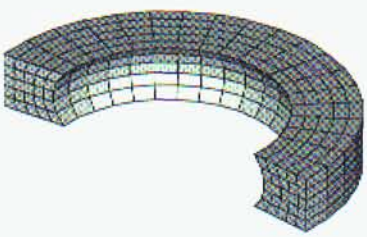


TRC-RCC subzoning

TRC-TRC subzoning



SPH-RCC subzoning



RCC-TOR subzoning

Most of these zones are defined entirely as the subtraction of one body from another. However, two of these zones require a third body to complete the definition of the input zone. The definition of the SPH-RCC requires the union with another RCC, but the dimensions of the RCC are dictated by the SPH and RCC desired. The second RCC must provide an extension of the two planes perpendicular to the axis of the subtracted RCC and must be large enough in radius to encompass all of the subzoned region. The planes of the RCCs need not be symmetric about the center of the sphere. The RCC-TOR will typically require the subtraction of another cylinder with radius equal to the torus radius of revolution and other parameters identical to the first RCC. (This is done to eliminate the hole in the center of the torus.) If employing one of these entities as an explicit subzone overlay, then only the two bodies need to be specified.

Automatic CAD Subzoning A method for subzoning CAD zones is provided. The user specifies the zone number that should be subzoned and the desired number of divisions in the x-, y-, and z-coordinates. ITS uses the bounding box of the CAD zone to create an RPP. This RPP serves as a subzone overlay for tallying purposes.

While this is a simple and automated subzoning technique, the resulting overlay may not be an efficient or reasonable subzoning scheme for the CAD zone. In some cases, a user may be able to manually specify a more efficient (nearly conformal) subzoning scheme for the CAD zone.

Subzone Overlays Subzone overlays (alternatively referred to as non-conformal subzoning) can be specified either implicitly or explicitly. The alternative is conformal subzoning. Conformal subzoning requires that the zone be defined by the same bodies as the subzoning scheme (with two exceptions stated in the description of multi-body subzoning that require an additional body to complete the zone description). Non-conformal subzoning requires that the zone be contained entirely within the subzone overlay. If this is not the case, errors can result due to invalid subzone indexes calculated during the transport process.

Implicit subzone overlays can be used by incorporating the subzone entity as the first body (or bodies) in the zone description. In this case, the overlay can be extracted from the zone description.

Explicit subzone overlays use different combinatorial descriptions for the input zone and the subzoning entity, and therefore use different geometrical descriptions for the transport process and for the subzone tallying process. In these terms, CAD subzoning is always explicit, but can be automated by having ITS convert the CAD bounding box into an RPP overlay.

11.1.4 Volume Specification

The volumes of the problem subzones must be specified through one of several automated or manual methods that are available. If automatic subzoning has not been requested for any input zone, the input zones are the same as the problem subzones. The volume of the escape zone is never specified. If the user wishes to supply the volumes for a run in which he has requested subzoning, he must ensure that the volumes are specified in the proper sequence.

A general scheme for the precise calculation of the volumes of zones defined by the combinatorial method is not possible. The user may select an option via [parameter(1)] associated with the GEOMETRY keyword. The default value of 0 will cause the code to set all volumes to 1.0 cm^3. A value of 1 allows the user to read in the volumes as described below. If the geometry is such that a satisfactory method exists for calculating the volumes internally, the user may set [parameter(1)] equal to 2 and use a code modification to insert the necessary logic at the proper place in Subroutine VOLACC.

A value of 3 triggers the code to automatically calculate subzone volumes and CAD zone volumes. The volumes of CG input zones that are not subzoned are set to 1.0 cm^3. Volumes can be obtained for some zones (the zones for which conformal subzoning is available) by requesting only 1 subzone (1 interval in each of the 3 dimensions). However, the user needs to be aware that these volumes are only available for the simple

single body or multi-body combinations for which subzoning is available. Attempts to calculate volumes for more complicated zones may result in non-conformal subzoning (see the discussion of Automatic Subzoning for more detail). A value of 4 for the first parameter on the GEOMETRY keyword has the same effect as option 3, except that the user may then overwrite subzone volumes for selected zones as described below.

If the value of [parameter(1)] is negative, the logic for setting subzone volumes will proceed based on the absolute value of the parameter, but printing of the volumes to the output file will be suppressed.

11.1.5 Material Specification

A material index is assigned to each input zone. A zero index defines a void zone; otherwise, the material indices are defined by the order in which the materials are specified in executing the cross-section generating code. The method of inputting the material indices is described under the keyword GEOMETRY.

11.2 Geometry Input Data

The geometry input for the ACCEPT codes is inserted according to the following sequence. The data is inserted in free format form with spaces or commas as delimiters. Simple examples can be found in its/Tests/RegTests/Input.

11.2.1 Body Data

The body data begin immediately after the line containing the GEOMETRY keyword. The method of describing each of the body types is discussed in the body definitions section and illustrated in Table 7. The description of each new body must begin a new line of input, and the first parameter on that line must be the appropriate three character code for the body type. Table 7 lists the additional input parameters required (no defaults) for each body type in their proper sequence. The user is free to distribute these parameters over as many lines as he pleases. A line with the keyword END signals that the description of all of the problem bodies is complete.

11.2.2 Input Zone Data

Geometrical specification of the input zones begins immediately after the line containing the END parameter for the body data. The method of describing the input zones in terms of the input bodies is discussed in Sec. 11.1.2. Body numbers are determined by the order in which the bodies are read in. The description of each new input zone must begin on a new line of input, and the first parameter on that line must be a character string beginning with the letter Z. Anything following this Z, that is not separated by a delimiter, is ignored. It is our convention to follow the Z with the zone number to improve readability for the user, but the code numbers zones in the order they are read regardless of the numbering after the Z. The Z parameter is followed by a string of parameters that specifies the input zone following the form of the right hand sides of the equations of Sec. 11.1.2. For example, input lines describing the two input zones in Fig. 14d are:

Z001 +1 +2

Z002 +3 -1 OR +3 -2

The user is free to distribute the parameters necessary for describing an input zone over as many lines as he pleases. A line with the keyword END signals that the description of all of the problem input zones is complete.

Body Type	Real Data Defining Particular Body					
BOX	Vx	Vy	Vz	A1x	A1y	A1z
	A2x	A2y	A2z	A3x	A3y	A3z
RPP	Xmin	Xmax	Ymin	Ymax	Zmin	Zmax
SPH	Vx	Vy	Vz	R		
RCC	Vx	Vy	Vz	Hx	Hy	Hz
	R					
REC	Vx	Vy	Vz	Hx	Hy	Hz
	R1x	R1y	R1z	R2x	R2y	R2z
ELL	V1x	V1y	V1z	V2x	V2y	V2z
	R					
TRC	Vx	Vy	Vz	Hx	Hy	Hz
	R1	R2				
WED	Vx	Vy	Vz	A1x	A1y	A1z
	A2x	A2y	A2z	A3x	A3y	A3z
ARB	V1x	V1y	V1z	V2x	V2y	V2z
	V3x	V3y	V3z	V4x	V4y	V4z
	V5x	V5y	V5z	V6x	V6y	V6z
	V7x	V7y	V7z	V8x	V8y	V8z
	Face Descriptions (see note below)					
TOR	Vx	Vy	Vz	Hx	Hy	Hz
	R	RH	R_p			
END	No Data					

Table 7. Data required to describe each body type

Note: The final line of the arbitrary polyhedron input contains a four-digit number for each of the six faces. Thirty data values are required for this body type; if there are fewer than eight corners and six faces, zero values must be entered.

11.2.3 Subzoning Data

The automatic subzoning capability is invoked in the following way. In the subzoning section of the GEOMETRY data, the keyword **SUBZONE** must appear followed by a parameter that specifies the number of the zone to be subzoned. If the CG zone description begins with the body number(s) that define the subzoning scheme (as either a conformal subzoning or an implicit non-conformal subzone overlay) or if one wishes to automatically subzone a CAD zone based on the RPP of the bounding box, then nothing else needs to appear on this line. To impose an explicit subzone overlay upon either a CG zone or a CAD zone, the keyword **OVERLAY** must be followed by a combinatorial description of the overlay scheme. Only 1 or 2 bodies are needed for the OVERLAY description. The multi-body subzoning descriptions should consist of one body minus another body.

The line immediately following must contain three integers that define the number of equal-increment subzones into which the zone is to be divided along the three coordinate directions. The three orthogonal directions corresponding to these three integers are defined in Sec. 11.1.3.

Additional information is required for multi-body subzoning and for azimuthal subzoning. For multi-body subzoning, there is a set of tertiary keywords defining various input zones to be subzoned that consist of the difference between two bodies. In this case, one of the following keywords must follow the three

subzoning integers: **RCC-RCC**, **RCC-TRC**, **TRC-TRC**, **TRC-RCC**, **SPH-SPH**, **TOR-TOR**, **SPH-RCC**, and **RCC-TOR**.

To perform azimuthal subzoning on any cylindrical, conical, spherical, or toroidal subzone entity, a reference direction is required to define the zero azimuth. The reference direction for a SPH or SPH-SPH is defined as the positive x-axis. A reference direction may be supplied for other subzoning entities (RCC, TRC, TOR, RCC-RCC, RCC-TRC, TRC-TRC, TRC-RCC, TOR-TOR, SPH-RCC, and RCC-TOR) by reference to another body. To specify such a reference direction, the input line with the three subzoning integers should contain the keyword **AZ**, followed by any body number except an RPP. The zero reference vector is the component of the V vector of the **AZ** body that is perpendicular to the H vector of the subzoned body. If that definition of the reference vector is null, the code attempts to use the +i, +j, and +k vector for the zero azimuth vector, in that order. If no **AZ** body is specified, the +i vector will be used as the body vector.

As an example, to impose an explicit non-conformal subzone overlay on input zone A in Fig. 14c (zone number 1) based on the surrounding RCC body (body number 3), one would specify:

```
SUBZONE 1 OVERLAY +3
  1 10 10
```

A line with the keyword END signals that the description of the problem subzoning is complete.

11.2.4 Volume Data

If [parameter(1)] associated with the GEOMETRY keyword is equal to 1, the array containing the volume data for the problem subzones is inserted immediately after the line containing the END parameter for the data specifying the subzoning schemes. The input zones are numbered according to the order in which they are read. If an input zone is not to be subzoned, then it becomes a single subzone. Therefore, if automatic subzoning has not been requested for any input zone, the subzones are identical with the input zones (except that the escape zone is not included among the subzones). If an input zone is to be subzoned, the subzones are numbered by incrementing the body-based coordinates in an order corresponding to the inverse of the order of the three integers specifying the subzoning (see previous subsection). For example, if an input zone consisting of an RPP is to be subzoned, the subzones are generated by first incrementing the z coordinate, then the y, and finally the x. The volume array must contain an entry for each problem subzone (no defaults), excluding only the escape zone.

If [parameter(1)] associated with the GEOMETRY keyword is equal to 4, then the code first calculates all subzone volumes analytically. These volumes can be overridden for subzones of user-specified zones. These appear immediately after the line containing the END parameter which terminates the input zone descriptions. The first line should give the index of the desired zone. The following lines should include the subzone index (a dummy value that is ignored) and the subzone volume (in cm^3). Any additional parameters will be ignored. There should be one line for each subzone, even for subzones that lie entirely outside of the original zone (that have zero volume). This format is repeated for each zone. The user can specify as many zones (that are subzoned, of course) as desired. This section is terminated with a line containing the keyword "END".

If [parameter(1)] is not equal to 1 or 4, these volume input data are omitted.

11.2.5 Material Data

For input of material data, return to the discussion under the GEOMETRY keyword.

11.3 Conventions for Escaping Particles

Any particle entering the is assumed to have escaped. In CG, the user is responsible for constructing a geometry such that this is true. In CAD, the escape zone is automatically determined.

In forward mode, the ESCAPE-SURFACES keyword may be used to indicate the surfaces on which escape particle tallies are desired. In adjoint mode, the SOURCE-SURFACES keyword may be used to indicate surfaces on which escape adjuncton particles are to be tallied as sources contributing to the response. The default in either mode is to perform total escape tally (equivalent to a single surface surrounding the geometry) for all particles entering the escape zone. If using these keywords, the user is responsible for ensuring that the surfaces listed under these keywords are a complete description of escape surfaces in forward mode and are an accurate representation of the surface-source quantity desired in adjoint mode.

Last Modified: February 2, 2004

12 Suggestions for Efficient Operation

The general operational limitations on the member codes of ITS are defined by the scope of the keyword input. However, specific information on the array sizes is provided by the Monte Carlo programs themselves. If while processing the keyword input, an array dimension required by a particular problem exceeds the default allocations as defined by Fortran 77 PARAMETER statements, the execution aborts immediately with a message to that effect being written to the output file. The user has the choice of either reducing the requirements or increasing the allocation. The latter is easily accomplished by modifying the relevant PARAMETER statement(s) in the its/Code/Hfiles/params.h file. The definitions of these integer parameters are located in the its/Code/Hfiles/defpar.h file. We do not wish to exaggerate the necessity of this procedure since, due to the number of code options and runtime options available, an optimum usage of memory is not likely to be the default configuration.

Immediately after a particular Monte Carlo member code has successfully processed the keyword input, it prints out an extensive comparison of the required array dimensions with the allocations as defined by the PARAMETER statements. If desired, a user may then customize the code to the problem by reducing all allocations to actual requirements. These modifications are optional, but may be required when running complex problems with the more complex codes on machines with limited fast memory.

WARNING: Care must be taken in reducing an allocation to zero since this may result in the upper bounds of the dimensions of certain arrays being set to zero; this will result in a fatal error since the lower bounds of the dimensions of all arrays is one.

Perhaps more important is the fact that the choice of certain input parameters can markedly affect the efficiency of the calculation; that is, the user's ability to obtain statistically meaningful output in a reasonable amount of time:

1. Obviously, the number of histories should be kept as small as possible. All member codes provide the user with estimates of the statistical uncertainties of the output data. Assuming that these uncertainties vary like the square root of the number of histories, these estimates then serve as a guide to the ultimate choice of the number of histories. The user must decide what level of statistical accuracy is acceptable for his or her particular application.

2. To achieve good parallel efficiency, the number of BATCHES should be an integer multiple of the number of the number of processors. The workload is distributed in batches. Having fewer batches than processors will result in some processors remaining idle. The first parameter of the TASKS keyword can be used to specify the number of processors or the code can determine the number of processors available.

3. The number of BATCHES should be at least 20 and should not be excessive. For a calculation running on hundreds of processors for many days, it is not excessive to have more than a thousand batches. However, there is communication, processing, and output overhead associated with beginning and ending batches and having too many can affect the efficiency.

4. Electron cutoffs should be as large as possible. For example, if the source is monoenergetic, a global electron cutoff equal to 5 or 10 percent of the source energy should be adequate. Because the logarithmic energy grid used in the electron transport technique becomes much finer at low energies, following electron histories down to low energies becomes very time consuming. On the other hand,

running time is not very sensitive to the value of the photon cutoff energy because low energy photons have a high probability of being absorbed after only a few interactions.

5. Similarly, electron trapping energies should be as large as possible. For example, consider the simulation of photoemission by low-energy photon sources. Because accurate simulation of boundary crossings is important, electron cutoffs must be low. On the other hand, if bremsstrahlung production is not important, as is likely in this case, electron trapping energies may be as high as the maximum source energy.

6. The requested energy, angle, and spatial resolutions should be no higher than necessary. Demanding excessive resolution only makes it more difficult – i.e., costly – to obtain statistically meaningful output.

Finally, the judicious use of a number of other variance reduction and options can markedly increase the efficiency of certain calculations. Specific examples of these are discussed under keywords BIASING, BIAS-GLOBAL, and BIAS-ZONE. Some of these are discussed in even more detail in the section on Biasing. Users are warned, however, that the reckless and indiscriminate use of biasing procedures can lead to misleading results. Any use of biasing schemes should be carefully considered and scrutinized.

Last Modified: March 17, 2004

13 Output Files

This section describes in detail the information present in output files. The output file can be divided into 3 possible sections: pre-processing information (anything preliminary to the Monte Carlo calculation), processing information (generated while the Monte Carlo calculation is performed), and results. There may not be any information generated while the Monte Carlo calculation is being performed. The output file is organized into sections designated by new carriage control pages ("1" in the first column). Each section of the output file is presented separately here. The meaning of each statement or value (and its units) that may appear in the output file is discussed. Many sections discussed here may not appear in a given output file, as the output depends upon both the code options and output options requested.

13.1 Pre-Processing Information

13.1.1 Output Header

The first information line conveys whether the continuous-energy code ("ITS") or the multigroup code ("MULTIGROUP ITS") is being executed. It also has the date of the code release. Unless you can be certain that you were using the specified code release, this date is not sufficient to reproduce a calculation, as the repository version of the code may have changed since the release without a corresponding change in this date.

The second line conveys the release version of the code.

The authors of the software are listed, and author contact information is provided.

The next section states the preprocessor directives used to obtain the executable with which the calculation was performed. This section does not state whether the MITS directive was used or not, as that information is conveyed in the first information line.

13.1.2 CAD Parameters *(CAD Only)*

If the CAD preprocessor directive is used, the flow logic integer requested for the calculation is stated with a table for determining the meaning of the flow logic integer. Also, the number of CAD zones that have been read is stated.

The CAD parameters requested (from the prmfile) are stated.

13.1.3 Reading Input

If the "ECHO 1" keyword is used, input is echoed to the output as it is read. This can provide the user with information about when and why an error occurs during the reading of input. However, for large input files, it may be desirable not to echo the input to the output file, so as to minimize the size of the output file. The entire input file is included in a job file, so this section may be redundant.

Minimal processing is performed while the input is being read, but some processing information may be reported in this section as it is performed. An example of this is (for **ACCEPT**) information on subzone volumes and how they were obtained. For **CAD** calculations, negative volumes indicate that the volume of the subzone intersected with the CAD zone and the volume of the CG subzone were in agreement. This means that the subzone lies entirely within the zone, and therefore CG-based electron trapping logic may be activated in that subzone. The negative values are merely informative, and positive values are used for all result normalizations.

Error and warning messages may be included in this section. These messages are generally preceded by ">>>>>" and include the words "ERROR" or "WARNING".

13.1.4 Reading Cross Section Data

For ITS (not **MITS**), this section includes:

- The title of the XGEN calculation that produced the cross section file.
- The number of cross section sets available in the cross section file is printed. This is the number of unique materials for which data was generated using XGEN.
- For each cross section set in the file the following parameters (used by XGEN to produce the cross sections) are stated:
  - The density of the material in g/cm^3.
  - The "detour" of the material. This is the ratio of the practical range to continuous-slowing-down-approximation range for an electron at the maximum energy of the set.
  - The "I(BL)" factor is Seltzer's empirical modification to the Blunck-Leisegang formulation for sampling from a truncated collisional energy-loss straggling distribution for electrons.
  - For each element in the material:
    * Z is the atomic number of the element.
    * A is the mean atomic weight of the element.
    * W is the weight fraction of the element in the material.
  - ITRM is the level of data contained in the cross section file. If less than 5, ITS cannot be run.
  - ISGN is the cross section model: 1=Mott Electron, 2=Mott Positron, 3=Screened Rutherford Electron, 4=Screened Rutherford Positron. This should always be 1, unless the XGEN code has been modified.
  - ISUB is the number of electron substeps taken per step in the condensed history algorithm. If the number of substeps per step is allowed to vary across the energy grid, then ISUB applies only to the first energy span. The XGEN default is for ISUB to remain constant across the energy grid, but see INDEX/JSUB under the DATAPREP DATA for more information.
  - INAL is the option used in XGEN for calculating eta in the Mott elastic cross section.
  - ICYC is the option used in XGEN for generating the electron energy grid. ICYC=1 means that a logarithmic scheme has been used (see the next parameter), and this should always be the case unless the XGEN code has been modified.
  - NCYC is the parameter determining the spacing of the electron energy grid. Successive energies are related by $E_{i+1} = 2^{-1/NCYC} E_i$.
  - NMAX is the number of electron energy grid values.
  - EMAX is the maximum energy in MeV of the electron energy grid.
  - EMIN is the minimum energy in MeV of the electron energy grid.
  - RMAX is the maximum electron range in g/cm^2 in the material.
  - LMAT is the number of elements in the material.
  - MMAX is the number of angular bins in the multiple scattering distribution.

  - The value of INDEX is stated. This is a three digit number. If the first digit IDST is 1, then the cumulative multiple scattering distributions are to be read into ITS from the cross section set. If the second digit IAVE is 2, then the average cosines for the multiple scattering distributions are to be read. If IAVE equals 1, then ITS cannot be run. If the third digit JSUB is 2, then the number of substeps per step for the condensed history algorithm is allowed to vary across the energy grid, in which case the number of substeps is read from the cross section set.

- A list of the data sets read from the cross section file for each material.

For **MITS**, this section includes:

- The title of the CEPXS calculation. (This is always "mits data file".)
- The number of materials for which data is available in the cross section set. This is the number of unique material compositions for which data was generated using CEPXS.
- The number of unique materials (both composition and density) labelled as material-densities. This is greater than or equal to the number of unique material compositions. (Some data is identical for materials with the same composition but different densities, and less memory is required by taking advantage of this.)
- The number of energy groups for all species.
- The particle species coupling scheme used to generate the cross sections.
- If neutrons are included in the cross sections, the neutron energy group structure in MeV.
- If electrons are included in the cross sections, the electron energy group structure in MeV.
- If photons are included in the cross sections, the photon energy group structure in MeV, and information about fluorescence lines, if any are included in the photon cross sections. Fluorescence information includes the element and shell-transition generating the line.
- For each material:
  - The density of the material in g/cm^3.
  - The "detour" of the material. This is the ratio of the practical range to continuous-slowing-down-approximation range for an electron at the maximum energy of the set.
  - For each element in the material:
    * Z is the atomic number of the element.
    * A is the mean atomic weight of the element.
    * W is the weight fraction of the element in the material.
- The scheme used to generate Fokker-Planck scattering-angles. The scattering angles may be energy- and material-dependent to mimic ITS substeps, or they may be constant. Either way, the maximum scattering angle will be written to the output (as the cosine of the angle).
- The treatment of positrons. Positrons may be tracked, not tracked, or treated as electrons. In the last two cases, annihilation will occur at the pair interaction site.
- If photons are included in the cross sections, the group into which annihilation radiation is produced, and whether that group is a line group.

13.1.5 Processing Input

This section will only contain information if errors or warnings are generated while processing input and cross section data. Checks are performed for inconsistencies and other errors in input keywords, cross section data, energy ranges, etc. These messages are generally preceded by ">>>>>" and include the words "ERROR" or "WARNING". A warning is generally generated instead of an error if an inconsistency has been resolved by the code in such a way as to allow the calculation to proceed. The user should always check for such warnings to determine if the calculation performed was the desired calculation.

13.1.6 Storage Requirements vs. Allocations

This section compares the size of array dimensions required with the allocation for those arrays. The user may be able to decrease some array allocations to cope with memory constraints. Generally, an error will have been generated before this point if an array has insufficient allocation. Array dimension parameters are specified in its/Code/Hfiles/params.h.

13.1.7 Geometry-Dependent Input

Geometry-dependent biasing settings and material assignments are listed for each zone.
For **TIGER** and **CYLTRAN**, the spatial extent of each zone is then listed.

13.1.8 Source Information

The energy, spatial, and directional distribution of the source is stated. In adjoint, this is the distribution of the adjuncton source (as specified by the DETECTOR-RESPONSE).

13.1.9 Output Options

The number of batches and histories per batch are stated.

Optional output requests are stated here. For each differential quantity requested, the binning structure is stated. For example, if ELECTRON-FLUX is requested, then the energy, polar-angle, and azimuthal-angle binning structure for the electron flux is stated. Keywords that will trigger binning information to be printed here are: ELECTRON-, PHOTON-, or NEUTRON- in combination with -ESCAPE, -FLUX, -SURFACE-SOURCE, or -VOLUME-SOURCE, and PULSE-HEIGHT.

13.1.10 Physical Options

The options used for modeling physics are listed. This includes physics for which input keywords provide switchs or cross section scaling, and physics that may or may not be included, such as fluorescence lines and positron annihilation.

13.2 Monte Carlo Output

13.2.1 Parallel Processing *(MPI Only)*

The following information is printed before the Monte Carlo calculation begins:

- Whether the load distribution is static or dynamic. This is directly related to use of the **DYNAMIC** preprocessor definition.
- Number of processes. This is the number of processors determined to be available.

- Master option. If equal to 1, then the master process performs Monte Carlo batch calculations. If equal to 0, then the master does not perform Monte Carlo calculations. The latter is always the case for dynamic load balancing.
- Number of tasks (requested). This is the number of processors that the user requested.
- Number of tasks (adjusted). If the number of processors requested was greater than the number of processors available, then the number actually used is stated.
- Intermediate print. This is the frequency (in number of batches) with which the intermediate output will be written.
- Allowed time factor. This number (if greater than zero) is the factor by which batch times are allowed to deviate from the average batch time. That is, if a batch requires more time than the average batch time multiplied by this factor, then the run is terminated.
- For each message broadcast from the master to all subtasks, the variable name at which the message starts and the length of the message (in bytes).

The following information is printed during the Monte Carlo calculation:

- For static calculations (not **DYNAMIC**) at the start of each cycle, the number of tasks performing Monte Carlo calculations and the initial random number seed.
- The subtask number and the random number seed assigned to initiate a batch. This is printed for the master task (subtask 0) and for batches corresponding to the intermediate output print frequency.
- The task number, the number of random numbers used, and the batch time. This is printed when the master finishes a batch or when a batch, corresponding to the intermediate output print frequency, returns its results to the master.
- The first time a batch is broadcast from a subtask to the master, for each message to be sent, the variable name at which the message starts and the length of the message (in bytes).

13.2.2 Monte Carlo Errors

Errors that occur during the Monte Carlo calculation will appear before the start of the results. This information does not have a section header. The information will appear to be at the end of the physical options section, or for **MPI** it will appear to be part of the parallel processing section. These messages generally are preceded by ">>>>>", include the word "ERROR", and list information that may be useful for debugging the error.

13.3 Results

Except in the initial diagnostics tables containing accounting information, every output quantity is followed by a one- or two-digit integer that is an estimate of the one-sigma statistical uncertainty of that quantity expressed as a percentage of the quantity. Details of the method used to obtain these statistical data are given in the Statistics section.

Almost all quantities in the results section are carried over during a dump and restart. Exceptions to this are random number information, that is specific to the latest batch or cycle, and some timing data that is specific to the latest batch or current run.

13.3.1 Diagnostics

Timing information:

- The percent of the problem completed. In the output file, this states that the problem is 100 percent complete, unless the PRINT-ALL keyword is used. This can be a useful marker while examining intermediate results to determine the progress of a calculation, either in the intermediate output file or in the output file if the PRINT-ALL keyword is used.
- In serial if a timer is available, the estimated time to finish is printed. This is useful in examining intermediate output, while a calculation is proceeding.
- In serial if a timer is available, the average time per batch.
- In parallel, the Monte Carlo time for the current batch. In the output file, this will be the last batch to complete (unless the PRINT-ALL keyword is used).

Random number seed and usage information:

- The initial random number seed of this cycle. In serial, a cycle is the entire run. For MPI with static load balancing, a cycle is one set of batches performed in parallel. For example, to perform 20 batches on 5 processors requires 4 cycles of 5 batches each. For MPI with dynamic load balancing, a cycle has no meaning, and the random number seed reported is the seed for the next batch.
- The initial random number seed of this batch. In serial, this batch is the last batch calculation performed. For MPI, this batch is the last for which a set of results is received by the master task.
- The initial random number seed for the next batch. This is the seed that would be used if the calculation continued to perform another batch. This seed will be used if a restart is performed. This seed may be used with the RANDOM-NUMBER keyword to initiate a calculation using an independent series of random numbers.
- The number of random numbers generated in this batch.
- Cumulated number of random numbers generated. This is the total number of random numbers used in the current run. This counter is reinitialized when the NEW-DATA-SET keyword is used, but it will be maintained through a restart.

Average source energy in MeV. In adjoint, this refers to the adjuncton source. This includes tallies for the energy of source particles that are determined to have energies below the cutoff energy that are not tracked.

For **MCODES**, a list of tallies precedes the standard diagnostics tables. These are mostly self-explanatory. The "corner problem" arises in some cases when a particle's curved trajectory in a field would go around a corner and approach a surface tangentially. An algorithm for coping with the corner problem has been implemented, and the counter records the number of times the more computationally expensive algorithm is invoked.

For ITS (not **MITS**), there are two diagnostics tables. The history table reports the number of events simulated:

- PRIM – primary histories simulated.

- SEC –secondary electron histories simulated. This is the sum of KNOCK, P E, PAIR, COM, and AUGER. An electron history is not simulated unless the electron is produced above the cutoff energy and is not immediately trapped.
- KNOCK – knock-on electrons simulated.
- P E – photo-electrons simulated.
- PAIR – pair-production electrons and positrons simulated.
- COM – Compton-produced electrons simulated.
- AUGER – Auger-produced electrons simulated.
- BREM – bremsstrahlung photons simulated.
- RAD – unscaled bremsstrahlung events sampled to account for radiation energy-loss straggling of electrons.
- XRAY – fluorescence photons simulated.
- REJ. LAND – number of times that sampling of electron energy-loss straggling was rejected, either to preserve the mean energy loss or because the energy loss for the step would have been greater than the initial electron energy.
- REJ. PEAL – number of times that sampling of the photo-electron emission angle yielded a scattering cosine with an absolute value greater than one. (The scattering cosine was set to 0.99999.)
- PRIM STEPS – condensed history steps simulated for primary electrons.
- SEC STEPS – condensed history steps simulated for secondary electrons.
- NBLK – number of times that sampling of electron energy-loss straggling would have resulted in electron energy gain. (Energy loss was set to zero.)
- INCOH. SCAT – incoherent (Compton) photon scattering events.
- COH. SCAT – coherent photon scattering events.

The second ITS table reports the number and energy of secondaries produced. This includes particles produced below cutoff that are not tracked, except in the case of knock-on electrons. The ENERGY is the mean energy in MeV of this type of secondary produced per primary history. The AVE ENERGY is the average energy in MeV of this type of secondary produced per interaction that produced this type of secondary. The NUMBER/PRIMARY is the estimated mean number of secondaries of this type that would be produced per primary history in an unbiased calculation (i.e., without cross section scaling). The NUMBER GENERATED is the number of production events simulated by the code.

- FIRST KNOCK – knock-on electrons produced by primary electrons.
- TOTAL KNOCK – knock-on electrons produced by all electrons.
- PHOTO-ELECTRON – photoelectric interactions producing electrons.
- PAIR – pair production interactions producing electrons and positrons.

- COMPTON – incoherent Compton scattering events imparting energy to electrons.
- AUGER – Auger electrons produced by either photon or electron interactions.
- FIRST BREMSSTRAHLUNG – bremsstrahlung photons produced by primary electrons.
- TOTAL BREMSSTRAHLUNG – bremsstrahlung photons produced by all electrons.
- X-RAY (P-IONIZATION) – fluorescence x-rays produced by photon interactions. For standard codes (not **PCODES**), this applies only to K-shell fluorescence x-rays.
- X-RAY (E-IONIZATION) – fluorescence x-rays produced by electron interactions. For standard codes (not **PCODES**), this applies only to K-shell fluorescence x-rays.
- ANNIHILATION QUANTA – photons produced by positron annihilation.

For **MITS**, the history diagnostic table reports the number of events simulated. All tallies disregard non-unity particle weights that may arise due to biasing.

- Primary histories – primary particle histories initiated. (In adjoint, these are adjuncton source particles and may include a mix of particle species.)
- Unscattered primary photons – photons that do not interact from initiation to escape.
- Electron escape scores – all electrons entering the escape zone through any surface.
- Photon escape scores – all photons entering the escape zone through any surface.
- Neutron escape scores – all neutrons entering the escape zone through any surface.
- Cutoff scores – all particles slowing down (in adjoint, speeding up) past a cutoff energy.
- Boltzmann and FP interactions – any type of scattering or absorption interaction with the medium. The breakdown of these interactions follows with descriptions indented.
  - FP interactions – electron Fokker-Planck angular scattering interactions.
  - Electron elastic interactions – electron angular scattering interactions without correlated energy loss.
  - Photon coherent scattering – photon angular scattering interactions without energy loss.
  - Electron inelastic interactions (non-absorption) – an electron interaction from which a particle emerges, other than Fokker-Planck or elastic scattering. As the transport is performed, it is possible to have an electron interaction produce a photon but not an electron, since the particles emerging from an interaction are sampled without preserving of the identity of the original particle.
  - Photon inelastic interactions (non-absorption) – a photon interaction from which a particle emerges, other than coherent scattering.
  - Neutron inelastic interactions (non-absorption) – a neutron interaction from which a particle emerges.
  - Electron inelastic interactions (absorption) – an electron interaction from which no particles emerge.
  - Photon inelastic interactions (absorption) – a photon interaction from which no particles emerge.

– Neutron inelastic interactions (absorption) – a neutron interaction from which no particles emerge.

- CPSL too small for scor/cpsl – In adjoint only, if an isotropic surface-source is desired, this tally records how many times an escaping adjuncton had an angular cosine too small to allow for the necessary angular weighting of the score. Currently, the angular cosine limit is 1×10^{-13}.

- Non-physical upscatters (in adjoint, non-physical downscatters) – If cross sections are generated using differing energy group structures for different particle species, then it is possible to have a cross section for producing particles in one group (A) from another group (B) that extends lower in energy. If a particle has an energy in group B below the lower energy bound of group A, and the sampling of an interaction produces a particle in group A, then the particle produced must have a higher energy than the interacting particle that produced it. This is non-physical. Such interactions are allowed, but they are tallied so that the user may know how many such interactions occur in a simulation.

- Exponent argument less than −88 in forcing logic – In **CYLTRAN** and **ACCEPT** only, if collision forcing is used for photons, the code may attempt to use an exponential argument less than −88. The value −88 is used instead to avoid underflow errors on some platforms. This parameter (defined as C88 in the Hfiles/params.h file) can be adjusted.

- Exponent argument less than −88 in sampling distance to collision – In **TIGER**, the problem described in the previous bullet can arise whether collision forcing is used or not.

- Exponent argument less than or equal to −88 in next-event logic, score was not tallied. If the NEXT-EVENT-ESCAPE keyword is used (or if the logic is used due to a PHOTON-ESCAPE or PHOTON-SURFACE-SOURCE request), then scores are not be tallied for which the probability of escape is small (and may cause an underflow error on some platforms). Note that because the tally is proportional to the probability of escape which is small, this is unlikely to affect results. The C88 paramater can be modified to determine the effect. Using a value greater than −88 (C88 less than 88) may result in a speed up, since ray-tracing is halted for any escape path when the probability falls below −C88.

- Exponent argument greater than −88 in next-event logic, score was tallied. This tally is the complement of the previous tally. If next-event logic is used for photon escape, then the sum of these two tallies should equal the number of photon escape scores (unless there have been lost particles during the next-event logic).

The number of source particles and total energy in MeV of source particles rejected for being below the cutoff energy. (In adjoint, this means adjuncton source particles.) Because energy spectra are allowed to extend below global cutoff energies, this tally allows the user to assess the significance of source particles that were not tracked and whether the cutoff energy must be lowered to achieve an accurate simulation.

For the **PCODES**, a table of the number of electron impact ionizations is given for each shell, followed by a similar table for photoelectric ionizations .

13.3.2 Integral Electron Emission *(ITS ACCEPT Only, ELECTRON-EMISSION keyword)*

For each surface on which electron emission is calculated, the number and energy of electrons is reported. This is the average number per source particle and the average cumulative electron energy per source particle. Tallies are only recorded for electrons with energies above the electron cutoff energy. For each surface, a row in the table is labelled by the surface index and body index supplied by the user. The last row of the table reports tallies for any electrons emitted into the cavity through a surface not specified by the user. Any tallies in this row may be an indication of an error in the problem specification.

13.3.3 Integral Escape *(Forward Only)*

For a photon source, the number and energy escape fractions for unscattered primary photons is printed.

For ITS (not **MITS**), the number escape fraction (that is, number per source particle) is given for electron generated secondary electrons (E-SEC), photon-generated secondary electrons (P-SEC), and annihilation photons. The number escape fraction is given for x-rays generated in each material. For the non-**PCODES**, x-rays are only generated for the K-shell. This information is given for each surface requested or all surfaces by default (see the ESCAPE-SURFACES keyword for more information).

The next section contains tables of the number and energy escape fractions for each particle species. For all codes, information is given for electrons and photons. For **MITS**, information is also given for positrons and neutrons. For ITS, positrons are counted as electrons here. This information is given for each surface requested or all surfaces by default (see the ESCAPE-SURFACES keyword for more information).

For ITS, the following values are stated:

- Energy escape fraction below cutoff. These electrons and positrons are not included in any other output tallies.
- Net charge escape fraction below cutoff. These electrons and positrons are not included in any other output tallies.
- Net charge escape fraction above cutoff. This quantity may be different than the number escape fraction given for electrons in the above table, because this quantity assigns a negative weight to escaping positrons.

The energy of cutoff photons assumed to have escaped is stated. This quantity should be zero, unless the CUTOFF-PHOTONS-ESCAPE keyword has been used.

13.3.4 Boundary Currents *(TIGER Only)*

The electron boundary currents at each material interface are reported. For both transmission (particles going in the positive-z direction) and reflection (particles going in the negative-z direction), the number currents and energy currents are listed with the percent statistical uncertainty. The currents will include electrons below the cutoff energy, unless the NO-KICKING keyword is used.

Positrons are scored with electrons in this tally, in the same way and also with positive weights.

13.3.5 Energy and Charge Deposition *(Forward Only)*

A new section is initiated for every 50 subzones, and header information is repeated, unless one of the NO-DEPOSITION-OUTPUT or NO-SZDEPOSITION-OUTPUT keywords are used. Results are normalized for a single source particle. Besides reporting total energy and charge deposition, three subdivisions are given for each. For ITS, the subdivisions are based on the physics resulting in the deposition. For MITS, the subdivisions are based on the mechanics of the code. For ITS (not **MITS**), the four columns are PRIM, E-SEC, P-SEC, and TOTAL.

- PRIM is deposition directly by the primary particle. For photon sources the PRIM energy deposition should be zero, since only electrons deposit energy.
- E-SEC is the redistribution of energy or charge due to the secondaries of an electron (or positron), i.e., knock-on electrons or Auger electrons produced by electron (or positron) ionization. In ITS, the primary elecced below the cutoff energy and therefore not tracked. As with the E-SEC tally, it is not uncommon to find negative energy deposition tallies for P-SEC, since this is a redistribution tally.

- TOTAL is all deposition in the subzone. This should equal the sum of the other three.

For **MITS**, the four columns are MICRO, TRACK, FLCUT, and TOTAL.

- MICRO is deposition due to microscopic deposition tallies. This should only be non-zero if the MICRO keyword is used.
- TRACK is deposition due to flux-fold tallies. If using the MICRO keyword, this is only due to continuous-slowing-down of electrons.
- FLCUT is deposition by particles falling below the cutoff energy or electrons being trapped.
- TOTAL is all deposition in the subzone. This should equal the sum of the other three.

For **TIGER**, the energy deposition section(s) appears before the charge deposition section(s). Energy deposition is given in MeV$\times$cm^2/g. Charge deposition is in electrons$\times$cm^2/g. The columns for both sections report the subzone number, the material number for the subzone, the minimum-z edge of the subzone in cm, the maximum-z edge of the subzone in cm, the minimum-z edge of the subzone in g/cm^2, the maximum-z edge of the subzone in g/cm^2, and the four deposition values (with percent uncertainties). At the end of each table is a row of totals across all subzones in the problem.

For **CYLTRAN**, the energy deposition section(s) appears before the charge deposition section(s). Energy deposition is given in units of MeV, and charge deposition is given in units of electron charge (i.e., one electron deposited a charge of 1.0, and one positron deposited a charge of -1.0). The columns for the energy deposition table report the subzone number, the material number for the subzone, the mass of the subzone in grams, the volume of the subzone in cm^3, and the four deposition values (with percent uncertainties). The columns for the charge deposition table report the subzone number, the material number for the subzone, the minimum-z edge in cm, the maximum-z edge in cm, the inner radius edge in cm, the outer radius edge in cm, the lower azimuthal boundary in degrees, the upper azimuthal boundary in degrees, and the four deposition values (with percent uncertainties). At the end of each table is a row of totals across all subzones in the problem.

For **ACCEPT**, energy and charge deposition information are contained in the same table. Unless the DEPOSITION-UNITS keyword is used, energy deposition is given in units of MeV, and charge deposition is given in units of electron charge (i.e., one electron deposited a charge of 1.0, and one positron deposited a charge of -1.0). The first column contains the subzone number. If subzoning is not activated, the zone number is the same as the subzone number. If subzoning is activated in the problem, the start of each zone is marked by a line stating the "INPUT ZONE NUMBER", and for the zones that are subzoned, an additional line states the total deposition quantities for the zone and the following line notes how the subzone numbers are incremented within the zone. The second column states the material number for the subzone. The remaining columns report the four deposition values (with percent uncertainties) for energy deposition and then for charge deposition. At the end of the table is a row of totals across all subzones in the problem. The totals are not normalized by the DEPOSITION-UNITS scaling factors, so they are always in units of MeV for energy deposition and electron charge for charge deposition.

The energy conservation fraction is the sum of all energy accounted for by escape and deposition divided by the average source energy. For **TIGER** and **CYLTRAN**, this quantity appears immediately following the energy deposition table.

The charge conservation fraction for an electron source is the sum of all charge accounted for by escape and deposition in units of electron charge. The charge conservation fraction for a photon source is one minus the sum of all charge accounted for by escape and deposition. For either source, this statistical quantity should converge to one.

13.3.6 Particle Flux *(Forward Only)*

All flux estimates are obtained via a track-length tally and are normalized to one source particle, the volume of subzone, the energy interval (for energy differential tables), and the angular interval (for angle differential tables). The volume used for this normalization may not be the actual volume of the subzone, depending upon the geometry volume option requested. Unless the "GEOMETRY -3" flag is used, the volumes of all subzones are printed in the Reading Input section of the output (see section 13.1.3), and these volumes are used for the normalization. In **TIGER**, the energy spectrum flux is stated in units of #/MeV, the energy spectrum and angular distribution flux is stated in units of #/MeV/sr, and total angular distribution flux is stated in units of #/sr. In **CYLTRAN** and **ACCEPT** (if the subzone volumes are accurate) the energy spectrum flux is stated in units of #/cm^2/MeV, the energy spectrum and angular distribution flux is stated in units of #/cm^2/MeV/sr, and total angular distribution flux is stated in units of #/cm^2/sr. WARNING: In **ACCEPT**, automatic subzone volume calculation may not accurately describe the intersection of subzones with the zone when nonconformal subzone overlays are used. In **CYLTRAN** and **TIGER**, subzone volumes are always accurate because of the simplified geometries.

Electron *(ELECTRON-FLUX keyword)* For ITS (not **MITS**), the "electron left at flux cutoff energy" is listed before the energy spectrum of the electron flux. This is given as sets of two rows of numbers. The first row contains the subzone numbers. The second row (in columns aligned with the subzone numbers) contains the number of electrons per source particle left at the flux cutoff energy in each subzone and the associated percent uncertainty. Each row contains up to ten subzone quantities.

A table of "energy spectrum of electron flux" appears for each subzone of the zones requested with the ELECTRON-FLUX keyword. For each energy interval requested, the electron flux and percent uncertainty are stated.

A table of "energy spectrum and angular distribution of electron flux" is given for each subzone of the zones requested. For **ACCEPT** and **CYLTRAN**, the azimuthal interval is stated in the header for each table. For **TIGER**, the azimuthal interval is always implied to be 0 to 360 degrees, since it is a one-dimensional code that can not resolve the azimuthal direction. Column headers state the polar (theta) interval in degrees for the corresponding header. (If the direction-sphere option is used, the azimuthal and polar information appear in the opposite locations.) The energy interval associated with the flux values is stated at the start of each row. The flux value and percent uncertainty is then stated for the energy interval given at the start of the row, the polar interval stated above the column, and the azimuthal interval stated in the table header.

At the bottom of each energy-angular flux table, the total angular distribution of flux is stated for the entire energy interval for which flux was tallied. (This total energy interval is stated at the start of the row.) Again, the angular intervals correspond to the polar interval in the column header and the azimuthal interval in the table header.

Photon *(PHOTON-FLUX keyword)* A table of "energy spectrum of photon flux" appears for each subzone of the zones requested with the PHOTON-FLUX keyword. For each energy interval requested, the photon flux and percent uncertainty are stated. The flux from continuum radiation and line radiation are listed separately. The continuum radiation appears in the same format as the electron flux values. At the start of each row containing line radiation information, the energy of the line is stated. Annihilation radiation (if applicable) appears first. Then, line radiation fluxes are given for each possible transition in each material. The transition producing the line radiation is stated with the line energy at the start of each row. (Only K-shell transitions are simulated in the standard ITS codes. The **PCODES** allow more detailed line radiation. For line radiation fluxes in **MITS** calculations, line radiation groups must be present in the cross section set, and therefore must be requested in the CEPXS input.)

A table of "energy spectrum and angular distribution of photon flux" is given for each subzone of the zones requested. The continuum and line radiation are stated separately. The table format is like the electron energy-angular flux, with the addition of line radiation as in the energy spectrum photon flux.

At the bottom of each energy-angular flux table, the total angular distribution of flux is stated for the entire energy interval for which flux was tallied. The total includes both continuum and line radiation.

Neutron ***(MITS Only, NEUTRON-FLUX keyword)*** The tables of "energy spectrum of neutron flux" and "energy spectrum and angular distribution of neutron flux" are given in the same formats as the electron flux tables.

13.3.7 Spectrum of Absorbed Energy *(ITS Only, PULSE-HEIGHT keyword)*

The energy intervals for recording tallies are listed at the start of each row of data. The number of tallies and percent uncertainties in the estimate follow. The results are in units of number of tallies per MeV, and are normalized to a single history (i.e., a single source particle).

It is common practice to have the first energy interval and last energy interval be very small. The first energy interval, spanning a range from the source energy to an energy slightly smaller than the source energy, records tallies for the total of absorption of source energy within the detector. The last energy interval, spanning a range from slightly larger than zero energy to zero, records tallies for no source energy absorption within the detector.

The spectrum of absorbed energy is a pseudo-pulse height distribution. It differs from a true pulse height distribution in that ITS is a Class I electron transport code, meaning that it does not correlate electron energy loss with the energy imparted to knock-on electrons. (Energy loss due to bremsstrahlung production is sampled with correlation to secondary production.) Electron energy loss is sampled based on straggling distributions to determine the energy lost by an electron. The energy lost is deposited in the medium. The production of knock-on electron is sampled separately. Energy initiating a knock-on electron is removed from the medium. In a given history, it is possible to have more energy removed from a subzone than is deposited, but statistically the energy deposition is accurate as the number of histories is increased. Pulse height distributions are tallies on a per-history basis. It is possible to have physically unrealistic negative absorbed energy tallies. A diagnostic following the absorbed energy table states the number of counts that were rejected due to negative energy deposition.

13.3.8 Electron Emission *(ITS ACCEPT Only, ELECTRON-EMISSION keyword)*

The first line of each header states that the table reports the energy spectrum and angular emission distribution of electrons. The second line states which surface of which body the electron emission has been tallied for, and if applicable also reports the subsurface index for that surface. Results are normalized for a single source particle and are given in units of #/MeV-sr (number of electrons emitted divided by both the energy interval and solid angle interval). On the fifth line, the azimuthal interval is stated (or polar interval, if direction-sphere binning is used).

Column headers state the polar (theta) interval in degrees for the corresponding header. (If the direction-sphere option is used, the azimuthal and polar information appear in the opposite locations.) The energy interval associated with the electron emission values is stated at the start of each row. The emission value and percent uncertainty is then stated for the energy interval given at the start of the row, the polar interval stated above the column, and the azimuthal interval stated in the table header.

13.3.9 Escape Spectra *(Forward Only)*

All escape estimates are obtained by tallying particles entering the escape zone. For **ACCEPT**, the accuracy of the escape tallies depends upon an adequately defined escape zone (e.g. it should be non-reentrant). Escape results are normalized to one source particle, the energy interval (for energy differential tables), and the angular interval (for angle differential tables).

Separate tables of escape information are given for each surface requested. For **ACCEPT**, the default is to report the escape information "through all surfaces" in integrated table(s). For **TIGER** and **CYLTRAN**, the default is to report escape information separately for each of the 2 or 3 possible escape surfaces. See the ESCAPE-SURFACES keyword for more information.

The energy spectrum escape is stated in units of #/MeV, the energy spectrum and angular distribution escape is stated in units of #/MeV/sr, and total angular distribution flux is stated in units of #/sr.

Electron ***(ELECTRON-ESCAPE keyword)*** For each surface requested with the ESCAPE-SURFACES keyword, two tables of data may be given depending upon the resolution requested with the ELECTRON-ESCAPE keyword.

A table of "energy spectrum of escaping electrons" is written first. For each energy interval requested, the electron escape and percent uncertainty are stated.

A table of "energy spectrum and angular escape distribution of escaping electrons" is presented next (if angular binning was requested). For **ACCEPT** and **CYLTRAN**, the azimuthal interval is stated in the header for each table. For **TIGER**, the azimuthal interval is always implied to be 0 to 360 degrees, since it is a one-dimensional code that can not resolve the azimuthal direction. Column headers state the polar (theta) interval in degrees for the corresponding header. (If the direction-sphere option is used, the azimuthal and polar information appear in the opposite locations.) The energy interval associated with the escape values is stated at the start of each row. The escape value and percent uncertainty is then stated for the energy interval given at the start of the row, the polar interval stated above the column, and the azimuthal interval stated in the table header.

At the bottom of each energy-angular escape table, the total angular distribution of escape is stated for the entire energy interval for which escape was tallied. (This total energy interval is stated at the start of the row.) Again, the angular intervals correspond to the polar interval in the column header and the azimuthal interval in the table header.

Escaping positrons are included in the electron escape tallies.

Photon ***(PHOTON-ESCAPE keyword)*** For each surface requested with the ESCAPE-SURFACES keyword, two tables of data may be given depending upon the resolution requested with the PHOTON-ESCAPE keyword.

A table of "energy spectrum of escaping photons" is written first. For each energy interval requested, the photon escape and percent uncertainty are stated. The escape of continuum radiation and line radiation are listed separately. The continuum radiation appears in the same format as the electron escape values. At the start of each row containing line radiation information, the energy of the line is stated. Annihilation radiation (if applicable) appears first. Then, line radiation escape is given for each possible transition in each material. The transition producing the line radiation is stated with the line energy at the start of each row. (Only K-shell transitions are simulated in the standard ITS codes. The **PCODES** allow more detailed line radiation. For line radiation escape in **MITS** calculations, line radiation groups must be present in the cross section set, and therefore must be requested in the CEPXS input.)

A table of "energy spectrum and angular escape distribution of escaping photons" is presented next (if angular binning was requested). The continuum and line radiation are stated separately. The table format

is like the electron energy-angular escape, with the addition of line radiation as in the energy spectrum of escaping photons.

At the bottom of each energy-angular escape table, the total angular distribution of escape is stated for the entire energy interval for which escape was tallied. The total includes both continuum and line radiation.

Neutron ***(MITS Only, NEUTRON-ESCAPE keyword)*** For each surface requested with the ESCAPE-SURFACES keyword, two tables of data may be given depending upon the resolution requested with the NEUTRON-ESCAPE keyword.

The tables of "energy spectrum of escaping neutrons" and "energy spectrum and angular escape distribution of escaping neutrons" are given in the same formats as the electron escape tables.

13.3.10 Sources and Responses *(Adjoint Only)*

In adjoint, a separate output section exists for each source energy spectrum. If no forward source spectrum is specified, results are based on the default flat energy spectrum (i.e., a unit strength flat energy spectrum is applied to each energy range for which results are reported). An output header is written for each set of output, and a new set of output is initiated if more than 8 columns are required. Each adjoint output header first describes the detector:

- Detector response type (DOSE, KERMA, CHARGE, ESCAPE ELECTRONS, ESCAPE PHOTONS, or ESCAPE NEUTRONS)
- Units of the response values (MeV-cm2/g-source-particle for dose or kerma, electrons-cm2/g-source-particle for charge, or number/source-particle for escape).
- For escape, the energy range of particles detected.
- For escape, the angular extent of particles detected.
- The location of the detector. (Surface for escape; point or volume for deposition.)
- For deposition, the material for the detector.

Next, the sources are described in a header section that applies to all detector-response values that follow:

- The location of the source (currently, only surfaces are available). For ACCEPT, the default is that the source is "through all surfaces" of the .
- The angular distribution assumed for the source (isotropic, cosine-law, or delta0-ave).
- The angular extent of the source in degrees. For energy intervals only, the source extends over the full angular range. For energy and angle intervals, separate headers may be provided for sources extending over only a portion of the azimuthal range (normally) or polar range (for direction-sphere).
- If results are normalized to one source particle, that is stated. Otherwise, the source strength is stated.

Finally, sources are described by row and column labels that apply only to detector-response values in the corresponding rows and columns:

- Row labels state the source energy ranges in MeV for each detector response value.
- Column labels state the polar range (normally) or azimuthal range (for direction-sphere) of the sources in degrees.

13.3.11 CAD diagnostics *(CAD, not CG_ONLY mode)*

Source particle location:

- Total calls – The number of times a source particle zone number was requested (should equal the number of histories).
- Percent right (mirroring only) – The percentage of agreement between CAD and CG.
- Percent wrong (mirroring only) – The percentage of disagreement between CAD and CG.
- Percent on boundary – The percentage of source particles for which the zone number could not be determined because the particle was located on a boundary.
- Percent unknown – The percentage of source particles for which the zone number could not be determined.
- Percent bad – The percentage of source particles for which CAD experienced a failure while trying to determine the particle location.
- Percent rejected – The percentage of source particles rejected due to failure to the determine zone number.

Distance to boundary:

- Total calls – The number of times the distance to a boundary was requested.
- Percent right (mirroring only) – The percentage of agreement between CAD and CG.
- Percent wrong (mirroring only) – The percentage of disagreement between CAD and CG.
- Percent no-hits – The percentage of attempts that failed to located a boundary in the direction requested.
- Percent rejected – The percentage of calls that resulted in a particle being rejected due to failure to find a valid distance to boundary. This is not the same as a percentage of source particles rejected, since some of these rejected particles may be secondary particles and may have a variety of weights due to biasing.
- 1 sidestep attempted – The number of times that, because a boundary could not be located, the particle was pushed 1×10^{-7} cm perpendicular to the direction of flight, and the distance to a boundary was attempted to be determined.
- 2 sidesteps attempted – The number of times that, because the first sidestep failed, the particle was pushed 1×10^{-7} cm perpendicular to the direction of flight and perpendicular to the direction of the first sidestep from the location of the particle before the first sidestep was attempted. The difference between this value and the "1 sidestep attempted" value is the number of times that the first sidestep succeeded and transport continued with the particle on a slightly altered path.
- 3 sidesteps attempted – The number of times that, because the second sidestep failed, the particle was pushed 1×10^{-7} cm perpendicular to the direction of flight (in the opposite direction of the first sidestep).

- 4 sidesteps attempted – The number of times that, because the third sidestep failed, the particle was pushed 1×10^{-7} cm perpendicular to the direction of flight (in the opposite direction of the second sidestep).
- Sidesteps failed – The number of times that a distance to boundary could not be determined either along the original particle path or along 4 different offset particle paths.

Boundary crossing location:

- Total calls – The number of times that the zone number of a particle location was requested after a particle crossed a boundary.
- Percent right (mirroring only) – The percentage of agreement between CAD and CG.
- Percent wrong (mirroring only) – The percentage of disagreement between CAD and CG.
- Percent on boundary – The percentage of particles for which the zone number could not be determined because the particle was located on a boundary.
- Percent unknown – The percentage of particles for which the zone number could not be determined.
- Percent bad – The percentage of particles for which CAD experienced a failure while trying to determine the particle location.
- Percent rejected – The percentage of particles rejected due to failure to determine the zone number.

13.3.12 Timing Data

All times are reported in seconds. Batch times record only the time during which the Monte Carlo calculation is being performed and exclude all I/O, pre-processing, and post-processing. In a RESTART calculation, only the cumulative Monte Carlo time, average batch time, and average history time include the times of the previous run(s).

With **MPI**, the function MPI_TIME is used to determine timing metrics for the calculation. This is a wall clock timer.

A timer may or may not be available for your platform. If a timer is not available, all timings will be reported as zero. If a timer is available, consult the Code Options section to determine whether the timer is a CPU or wall-clock timer.

- CAD preprocessing time (s) – This is only reported for **CAD** calculations. This is the time required to process the CAD geometry and input.
- Input processing time (s) – This is the time to read input and cross sections and perform preprocessing before initiating the Monte Carlo calculation.
- Output processing time (s) – This is only reported if not using **MPI**. This is the time required for post-processing and writing output, including intermediate output.
- Monte Carlo cycle time (s) – This is only reported if using **MPI** in static (not **DYNAMIC**) mode. This is the sum across all cycles of the longest batch time for each cycle.
- Longest Monte Carlo time (s) – This is only reported if using **MPI** in **DYNAMIC** mode. This is the longest batch time.

- Elapsed execution time (s) – This is the total run time for ITS. This includes I/O, preprocessing, Monte Carlo, and post-processing within the current run.
- Cumulative Monte Carlo time (s) – This is the sum of all batch times.
- Averaged Monte Carlo batch time (s) – This is the average time required to perform a batch calculation for all batch calculations performed.
- Average Monte Carlo history time (s) – This is the sum of all batch times divided by the number of Monte Carlo histories simulated.

Last Modified: January 9, 2004

14 P Codes

The SANDYL[14] code is a three-dimensional multimaterial code. Its construction was oriented toward relatively low-energy photon sources and the understanding of internal electromagnetic phenomena in complex geometries. In particular, it includes a detailed modeling of atomic shell ionization and relaxation phenomena for electron and photon energies down to 1.0 keV. The early codes of the TIGER series, on the other hand, were developed primarily for relativistic electron beam applications, where atomic shell effects usually play only a minor role and are, consequently, treated in a more cursory fashion. Nevertheless, these potential low-energy limitations were purely incidental, and there was no reason why the more complete description of atomic shell effects available in the SANDYL code could not be included in the standard codes of the TIGER series. Those codes in ITS that include the more detailed ionization/relaxation physics from the SANDYL code are referred to as the P codes. The P codes contain the logic necessary for describing ionization and relaxation of all K, L1, L2, L3, M (average), and N (average) shells having binding energies greater than 1.0 keV for elements with atomic numbers Z=1 to Z=100.[14][12]

Once a photoionization or electron impact ionization event has occurred, several different relaxation cascades are possible. A large quantity of atomic relaxation data is required for the stochastic description of these cascades. These data, together with the photoionization probabilities, are tabulated in Ref. [12], which also includes a discussion of the cross sections for electron impact ionization and details of the implementation of these processes in the P codes.

The standard codes only include a description of the electron impact ionization of the K shell of the highest-atomic-number element in a given material. Similarly, following either this impact ionization or a photoelectric event, these codes only model relaxation processes (production of Auger electrons and fluorescent photons) from this same shell. Nevertheless, for the vast majority of problems, the P codes give results that are virtually identical to those of the standard codes. This is important because the P codes require a significant increase in both memory and run time as compared with their standard counterparts. If the user feels that the additional sophistication of the P codes may be important for his or her application, results from the P codes should first be compared with those of the standard codes for sample problems typical of that application in order to determine whether the differences, if any, in the desired results are significant. This comparison should only be necessary for those applications where low-energy transport plays a major role – e.g., problems involving low-energy photon sources. Only if there are significant differences should the P codes be used for subsequent runs.

Last Modified: January 9, 2004

15 M Codes

In many instances the value of strictly collisional transport models is questionable because the actual experiments involve macroscopic electric and magnetic fields whose effects upon radiation transport not only cannot be neglected but may even be more important than the collisional effects. In order to address this situation, we have developed a model that combines sophisticated coupled electron/photon collisional transport with transport in externally applied macroscopic electric and magnetic fields of arbitrary spatial dependence.

The model allows magnetic fields in both material and void regions. Of course, magnetic fields alone will only deflect electrons without changing their energy. The procedure for combining collisional energy loss and deflection with magnetic deflection has been described elsewhere in detail.[15][33][34] Briefly stated, the rectilinear random-walk substeps of the field-free model[1][2] are replaced by numerically-integrated segments of field trajectories in vacua whose integrated areal densities are equal to those of the substeps. Sampled collisional deflections are superimposed upon the electron direction at the end of each of these vacuum-trajectory segments. The numerical integration scheme determines those locations along the segment that correspond to energy deposition and secondary production (knock-on electrons, bremsstrahlung photons and relaxation particles from electron impact ionization), as well as the intersections of the trajectory segments with material boundaries. Magnetic fields should be ignored in regions where transport is collision dominated, because the combined simulation is quite expensive in such cases, though the results are the same as for collisions alone.

Electric fields (or combined electric and magnetic fields) are allowed only in void regions. This restriction has been imposed because no sufficiently general scheme has been derived to account for the effects of changes in the electron energy produced by the macroscopic electric field upon the energy-dependent multiple-interaction collisional processes within a given substep. For those applications where electric fields are present within material media (e.g., potential buildup in dielectrics and sustaining fields in gas lasers), special algorithms[35] that depend upon the ratio of the electric potential gradient to the electronic stopping power must be introduced to handle this difficulty. Even with the restriction of electric fields to void regions, the model is applicable to a wide variety of problems – for example, problems involving accelerating diodes.

The method for accurately integrating the vacuum equations of motion in order to obtain the vacuum-trajectory segments is the essential feature of the model, whether the fields are present in material or in void regions. In voids the integration is interrupted whenever the trajectory intersects a material boundary or a problem-escape boundary. In material regions the integration is also and more frequently, interrupted whenever the areal density traversed corresponds to a location where energy is to be deposited or secondary production occurs, or equals the areal density of the appropriate substep. In the latter case, collisional scattering and energy loss are accounted for and a new trajectory segment is initiated A fourth- to fifth-order Runge-Kutta-Fehlberg routine with automatic step-size control (RKF)[16], substantially modified to include boundary-crossing logic and other constraints, is employed to integrate the equations of motion in vacuo. The reasons for this choice are discussed at length in Ref. [34].

The current algorithm includes a major improvement over the method described in Ref. [16]. The basic RKF integrator was designed to integrate over some specific interval of the independent variable. However, model applications invariably require interruption of the integration at the roots of any one of several possible constraint functions that are functions of the dependent variables. The most common example is the root corresponding to the intersection of an electron trajectory with a zone boundary. Other examples are the roots corresponding to the locations for energy deposition and secondary production. In the pre-ITS versions of the M codes, we were forced to use relatively crude approximate solutions at these roots, which limited the

overall accuracy of the model predictions to something substantially less than the inherent accuracy of the integrator.[16] The algorithm now includes an extended RKF procedure[17] that permits interruption of the integration at any one of a number of constraint functions of both the dependent and independent variables with an accuracy that is comparable to the inherent accuracy of the integrator. The more sophisticated user is free to add his own constraint functions for interrupting the integration. The constraint functions must be defined in subroutine CSTR.

Last Modified: February 2, 2004

16 Biasing Options and Variance Reduction

From a practical, if not theoretically rigorous, point of view, biasing in Monte Carlo can generally be described as the distortion of the natural analog to achieve variance reduction in certain desired output quantities. Variance reduction refers to the attainment of lower statistical uncertainty for the same amount of run time or, equivalently, the attainment of the same statistical uncertainty in a lesser amount of run time. Except where absolutely necessary, biasing should be used sparingly. In any case, it should be used with great care. Reckless use of biasing (overbiasing or underbiasing) can lead to results that are erratic and/or easily subject to misinterpretation. Nevertheless, there are biasing options in ITS that are easily accessed via input keywords and that have proven very useful in specific applications.

Zone-dependent particle cutoff energies can be specified with the CUTOFFS keyword. For other biasing options, there are two approaches available to the user. The BIAS-GLOBAL and BIAS-ZONE keywords may be used to set global biasing parameters and zone-dependent parameters, respectively. The BIAS-ZONE keyword is used for each zone in which a zone-dependent feature is desired and in some cases is needed to specify the zones in which global biasing settings are to be activated. Alternatively, the BIASING keyword may be used to simultaneously specify global biasing parameters, the zones in which global biasing parameters are to be applied, and zone-dependent biasing parameters.

16.1 Zone-Dependent Cutoff Energies

The user has the option of varying the cutoff energy for each species from zone to zone so long as the zone-dependent cutoffs are greater than or equal to the global cutoff energy for the particle species. The option is activated via the appropriate parameter associated with the CUTOFFS keyword.

For electrons in the continuous-energy codes, when the energy of the electron in a given zone falls below the cutoff for that zone, a check is first made to see if it is trapped in the sense described under subkeyword TRAP-ELECTRONS. If so, the history is terminated via on-the-spot deposition of charge and energy. Otherwise, except for the M codes, a final calculation of non-local energy and charge deposition is made based on the residual range of the electron (Subroutine KICK). For the M codes, a relatively low electron cutoff energy should be used because the history is always terminated via on-the-spot deposition of the charge and remaining energy of the electron. It is important to remember that there is no production of secondary particles by electrons below the zone-dependent cutoff, nor is there any contribution to electron flux and electron escape by such electrons.

For photons, electrons in the multigroup code, and neutrons, when the energy of a particle falls below the cutoff for the zone the particle is in, the history is terminated via on-the-spot deposition of charge and energy. Because photons can travel relatively large distances before interacting and photon transport is generally inexpensive compared to electron transport, this cutoff energy should be used with caution.

The zone-dependent cutoff option has proven useful in problems that involve the generation of bremsstrahlung in one region and deposition caused by that bremsstrahlung in another region. A relatively high cutoff may be used in the converter zone(s) since low-energy electrons are relatively inefficient for producing bremsstrahlung. On the other hand, in the zone(s) where deposition is dominated by bremsstrahlung transport, the user may be interested in the details of the deposition from the low-energy bremsstrahlung-produced secondaries (e.g., interface effects, in which case he may not want to kill those electrons with Russian Roulette; see also Sec. 16.3), or he may not want electron transport in those zones at all (bulk deposition; see also Sec. 16.5).

16.2 Forced Photon Collisions

An option is available for forcing a selected fraction of photons entering a given zone or leaving a collision site within a zone to interact in that zone. The option is activated via the COLLISION-FORCING subkeyword. The option is useful for forcing photons to interact in certain regions where their natural interaction probability is so small as to make it difficult to obtain statistically significant results. The values chosen for the forcing fractions must be > 0.0 and < 1.0, and are specified by the appropriate parameter for the given zone as described under the COLLISION-FORCING subkeyword. Care must be taken not to overbias. A forcing fraction of 1.0 will prevent any photons from escaping from the given zone and will prevent them from making contributions elsewhere in the results of the calculation (e.g., prevent them from contributing to the escape fractions).

16.3 Russian Roulette

When SCALE-BREMS and SCALE-EP options are used to increase the secondary photon population, it may be desirable to reduce the number of secondary electrons generated from the interaction of this artificially-high bremsstrahlung population. The ELECTRON-RR option allows the user to activate Russian Roulette in specified zones of the geometry. The default survival probability (when the feature has been activated) will return the secondary electron population to the naturally occurring number.

Although this procedure is very efficient for predicting external bremsstrahlung, it can lead to statistically poor results for the profiles of energy deposition, charge deposition, and electron flux in regions of the problem where these profiles are determined by the transport of and secondary electron production by the bremsstrahlung radiation. In such cases, the survival probability should be adjusted to ensure that a sufficiently large fraction of the secondary electrons are followed.

16.4 Next-Event Estimator for Photon Escape

For a geometry that is highly absorbent to secondary photons generated in the transport process, scoring as leakage photons only those secondary photons that actually escape the geometry while being tracked can be quite inefficient. (In the continuous-energy code, the contribution to total leakage from uncollided source photons is not scored because this contribution can usually be calculated analytically and might otherwise dominate the total leakage to the extent that the scattered contribution cannot be determined. However, a flag on the PHOTONS keyword can be used change this default.) This is remedied by using the next-event estimator for photon leakage. With this estimator, a score is obtained each time a photon emerges from a collision. The score is simply the emergent photon weight times the probability of escape without further interaction.

This option is automatically activated as a method of variance reduction for differential leakage when the PHOTON-ESCAPE keyword is used. Otherwise, the option is not used for the default prediction of integral leakage unless explicitly activated via the subkeyword, NEXT-EVENT-ESCAPE.

16.5 Photon Only Transport

In some calculations, electron transport is only important in a small portion of the geometry. In this case, it is more efficient to simply turn off electron production throughout much of the geometry rather than using a high electron cutoff energy. The PHOTRAN subkeyword provides this functionality. Zones in which electron transport is desired can be specified.

16.6 Scaling of Bremsstrahlung Production *(ITS Only)*

The user may artificially increase the bremsstrahlung production to improve the statistical accuracy of bremsstrahlung output without increasing the number of primary electron histories, which would be much more time consuming. The option is activated via the SCALE-BREMS subkeyword. The cross sections are scaled by a factor specified by the user.

Simultaneous scaling of the cross section for electron impact ionization probability is also desirable (see Sec. 16.8) and is performed by default when SCALE-BREMS is used. The material selected as the basis for scaling the impact ionization (the second parameter associated with the SCALE-BREMS subkeyword) should be that material which one would expect to dominate the bremsstrahlung production.

This option is used primarily for the prediction of external bremsstrahlung production (e.g., prediction of the environment of an x-ray source). Consequently, a Russian Roulette procedure may be desirable to reduce the number of secondary electrons generated. See Sec. 16.3 for details on the Russian Roulette feature.

16.7 Scaling of Electron-to-Photon Interactions *(MITS Only)*

Different types of interactions that have the same effect are not distinguished in the MITS codes. Therefore, it is not possible to simply scale the bremsstrahlung production of photons or the electron impact ionization production, as is done in the continuous-energy codes. Instead, the SCALE-EP feature allows the scaling of all photon-producing electron interaction cross sections. The cross sections of all materials are scaled by a factor specified by the user, but the scaled cross sections are only used in regions specified by the user.

16.8 Scaling the Probability for Electron Impact Ionization *(ITS Only)*

An option similar to that described in Sec. 16.6 permits the user to artificially increase characteristic x-ray production by scaling the cross section for electron impact ionization. This option is activated via the SCALE-IMPACT subkeyword. The cross section for electron impact ionization of each material is scaled so that an electron slowing down in that material from the maximum source energy to the global electron cutoff energy will, on the average, generate a number of ionization events equal to the value of the parameter associated with this subkeyword. A separate scaling factor is calculated for each material. Impact ionization scale factors are rounded to the nearest integer and are never allowed to be less than one. Impact ionization scaling is implemented in the same zones in which bremsstrahlung scaling is implemented.

If SCALE-BREMS is used and SCALE-IMPACT is not, electron impact ionization is scaled. The factor for this scaling is determined such that impact ionization events between the maximum source energy and the global electron cutoff energy equal 20% of the bremsstrahlung events over the same energy range in the material given by the second parameter on the SCALE-BREMS keyword. This number of ionization events is used to determine scale factors for each material.

16.9 Scaling of Photon-to-Electron Interactions *(MITS Only)*

The user may artificially increase the production of secondary electrons to improve the statistical accuracy of dose or charge deposition in a region of the problem rather than increasing the number of histories, which may be more time consuming if relatively few photons naturally interact in the region. This option is activated via the SCALE-PE subkeyword. The cross sections of all materials are scaled by a factor specified by the user, but the scaled cross sections are only used in regions specified by the user.

16.10 Trapped Electrons *(Forward Only)*

In certain problems where only electrons that cross certain boundaries are important, the option activated by the TRAP-ELECTRONS subkeyword may be employed to reduce run time significantly. The parameter associated with this subkeyword is the global electron trapping energy. In addition, zone-dependent electron trapping energies may be specified. Internally, an array of zone-dependent electron trapping energies is obtained, each element of which is the greater of the global trapping energy, the zone-dependent trapping energy, or the zone-dependent cutoff energy for that particular zone (see Sec. 16.1). The option becomes effective when an electron is trapped, that is, does not have enough energy to escape from a subzone. When an electron with energy less than the zone-dependent trapping energy is trapped, its history is immediately terminated via local (on-the-spot) deposition of its charge and remaining energy. This option is commonly used when one is primarily interested in the accurate transport of those electrons escaping from all or some portion of the problem geometry. Great care should be taken in employing this option where production of secondaries (e.g., bremsstrahlung) may be important because there is no secondary production by electrons whose histories are terminated in this fashion. It is important to note that the contribution to leakage from the subzone of any untrapped electrons with energies above the zone-dependent cutoff is much more rigorous than that of untrapped electrons with energies below the zone-dependent cutoff because a much cruder form of transport is employed for the latter. The decision as whether an electron is trapped or untrapped is based on subzone boundaries in the TIGER codes, axial and radial subzone boundaries in the CYLTRAN codes, and subzone (in subzoned regions) or code-zone (regions separated by an OR operator in the input-zone definitions and input zones defined without the OR operator; see the ACCEPT Geometry section) boundaries in the ACCEPT codes.TRAP-ELECTRONS

Last Modified: January 9, 2004

17 Statistics

A significant advantage of the ITS system is the computation of statistical uncertainties for virtually all output quantities. Under the default option, the total number of histories of primary particles are run in 20 equal batches. The output routine is called at the end of each batch. Immediately before each write statement, a call is made to Subroutine STATS. This routine (a) recalls the statistical variables from the previous batch corresponding to the output quantities about to be written, (b) computes the estimate of the statistical standard error (in percent) based on the number of batches that have been run, and (c) saves the statistical data from the current batch so that it will be available for the next batch. Unless the keyword PRINT-ALL is used, only the final results based on the total number of completed batches are printed out. The user may specify a number of batches other than 20 by using the keyword BATCHES as described in the Keywords for ITS section.

Under normal operation virtually every Monte Carlo output quantity is followed by a one- or two-digit integer from 0 through 99 (estimates even greater than 99 are shown as 99) that is the best estimate of the statistical standard error expressed as a percent of that output quantity:

$$(S.E.)_N = \frac{100}{|< x_N >|} \left| \frac{< x_N^2 > - < x_N >^2}{N-1} \right|^{1/2},$$

where

$$< x_N >= \frac{1}{N} \sum_{i=1}^{N} X_i,$$

and

$$< x_N^2 >= \frac{1}{N} \sum_{i=1}^{N} X_i^2,$$

The X_i's are the values of the quantity obtained from each batch, and N is the total number of completed batches (default 20).

Should the more sophisticated user wish to add additional tallies to any of the Monte Carlo member codes, he will find subroutine STATS to be a useful utility. STATS provides estimated statistical uncertainties for all output quantities. It has three formal parameters. The first is a temporary array containing the current batch values for the quantities for which statistical estimates are desired when the routine is called. The routine returns the cumulative batch averages in this same array for printing. The second parameter is an array that will return the statistical estimates for printing. The third parameter is the number of batch values to be processed with this call (the length of the first two parameter arrays).

Last Modified: January 9, 2004

18 Automatic Subzoning

Automatic subzoning refers to that feature of the ITS codes whereby the user may direct a particular member code to internally divide a given input zone into subzones for the purpose of obtaining the spatial variation of energy and charge deposition, electron flux, and photon flux within the given input zone. This powerful feature has the potential for substantially reducing (a) user time for generating the input file, (b) machine memory requirements, and (c) machine CPU time. Without this feature, the user would have to describe each subzone as a separate input zone. For example, a 10x10x10 subzoning of a rectangular parallelepiped with the ACCEPT codes would require at least 1000 lines of input without automatic subzoning. Because the subzones are defined in terms of equal increments of the intrinsic coordinates of the input zone, there is no need for explicit storage of the boundaries of subzones, and very little memory is required to locate the subzone containing an arbitrary point within the input zone. Finally, because the internal subzone boundaries are not material discontinuities, they can be ignored by the CPU-intensive tracking logic.

Automatic subzoning and related coding has grown with the development of the ITS system, and is still not as complete as we would like. In the TIGER codes, it was already virtually complete in Version 1.0. In that version of the CYLTRAN codes, there was some capability for pseudo-subzoning a solid annulus input zone in the radial and axial directions. However, the only reduction was in the sense of Item (a) above. Once these "subzones" were generated internally, they were treated like any other input zone. Their boundary information was permanently stored, and particles were tracked through them just like any other input zone. In Version 1.0 there was no subzoning of any kind in the ACCEPT codes. In Version 2.0, we implemented full automatic subzoning in the sense of Items (a), (b), and (c) above in the ACCEPT codes for input zones consisting of a single RCC or RPP body. In the latter we allowed three-dimensional subzoning, and in the former we allowed radial and axial subzoning. No additional automatic subzoning was implemented in Version 2.0 of CYLTRAN. In Version 3.0, we substantially extended this feature. We implemented full automatic subzoning in the CYLTRAN codes, extending it to three dimensions with the addition of azimuthal subzoning. Moreover, the ACCEPT codes featured the full three-dimensional subzoning of input zones consisting of a single body of type RCC, RPP, BOX, or SPH. Since Version 3.0, automatic subzoning has been added for input zones of a single body of type TRC, WED, and TOR. Also, a new type of subzoning has been added based on multi-body input zones. These are generally shells that consist of one body subtracted from the middle of another body. These entities are the RCC-RCC, RCC-TRC, TRC-TRC, TRC-RCC, SPH-SPH, TOR-TOR, SPH-RCC, and RCC-TOR.

There are, however, some aspects of the coding of ITS that do not yet take full advantage of or are not completely consistent with the philosophy of automatic subzoning. We discuss them here in terms of three basic questions that are frequently asked as a particle trajectory evolves within the problem geometry:

(1) Point location: In what input zone (or subzone) does a given point of the trajectory lie?

(2) Tracking: For a particle at a given point in a given input zone (or subzone) with a given direction, how far will it go before reaching the boundary of the given input zone (or subzone) if it continues moving in the given direction?

(3) Trapping test (electrons only): For an electron at a given point in a given input zone (or subzone), what is the minimum distance to the surface of that input zone (or subzone)?

In principle we would like the code to respond to such queries on a subzone rather than an input zone basis. In practice, it is felt that the overhead for doing the former may be so excessive as to negate the above mentioned advantages of automatic subzoning. We have taken a middle of the road approach based on our judgement of what is the best overall choice and what is feasible at this time. Since Version 3.0, subzone-based electron trapping (see the section on Biasing) has been extended to the ACCEPT codes and

to azimuthal subzones (in addition to the previously available axial and radial subzones) in the CYLTRAN codes.

In general, the features of the current ITS codes that use zones rather than subzones have to do with the actual scoring of certain spatially-dependent output quantities: energy deposition, electron flux, and photon flux. All of the scoring is subzone based, but there are alternative variations to what we currently use whose variance-reduction consequences have not been fully explored.

Energy deposition and scoring of volume-averaged electron flux are coupled. Both quantities are scored at the same point, as in Item (1) above, which is randomly selected along an electron random-walk substep, or partial substep if the substep encounters a material discontinuity. Because there is one score per substep, the use of automatic subzoning will not lead to variance reduction by increasing the number of scores per electron. Rather, the variance reduction from automatic subzoning is achieved because the time required to track an electron is reduced when subzone boundaries can be ignored.

There is, however, an alternative method of scoring these quantities, not yet implemented in ITS, that could potentially lead to further variance reduction in applications where the random walk substeps are greater than the subzone dimensions. We refer to this method as track-length apportioning. The segments of the substep within each subzone must first be determined. These segments times the electron weight are then scored as volume-averaged fluxes in the appropriate subzones. Similarly, the segments, as fractions of the total substep, are used to apportion the substep energy deposition among the subzones. Multiple scoring of these quantities per substep may lead to significant variance reduction. The caveat to this approach is that it requires what is equivalent to tracking, as in Item (2) above, among the subzones. It was the avoidance of tracking among subzone boundaries that was primarily responsible for the variance reduction achieved by automatic subzoning. Nevertheless, the more sophisticated user may use code modifications to implement this method. This is rather easily done for the TIGER codes (see discussion of photon flux below), but becomes progressively more difficult for the CYLTRAN and ACCEPT codes.

In the CYLTRAN and ACCEPT codes, scoring of volume-averaged photon flux is done in a fashion similar to that of electron flux. Here, however, instead of obtaining one score per substep, there is one score per free-flight photon trajectory segment between collisions or input- zone boundaries. Again, scoring is at a point, as in Item (1) above, that is randomly sampled along the free-flight segment. Automatic subzoning does not increase the number of scores, and variance reductions again derive from the avoidance of tracking among the subzone boundaries. However, when the dimensions of the subzones are much smaller than the average free-flight photon trajectory segments, there could actually be a variance increase relative to the variance that would have been obtained had each subzone been defined explicitly as an input zone, since the number of scores is greatly reduced in the former case. Whether there is an overall variance increase or reduction is, of course, very problem dependent.

Because photon mean free paths tend to be much larger than electron random-walk substeps, and because one-dimensional tracking is relatively simple, it was decided to apportion the free-flight photon trajectory segments among the subzones in the TIGER codes when scoring photon flux. Thus, significant additional variance reduction, over and above that from automatic subzoning alone, may accrue to the estimation of photon fluxes with the TIGER codes, unless subzone boundaries tend to be much larger than photon mean free paths. Using code modifications, this logic may easily be extended to the calculation of electron flux in the TIGER codes.

18.1 Non-Conformal Subzone Overlays

At times, subzoning is desired for a zone that is quite complicated. For this purpose, a more general method is provided based on overlaying the zone with the subzoning of a simple entity. An input zone with arbitrary shape (that may include unions or CAD geometry zones) can be subzoned by first completely enclosing it in any simple body or one of the shells available for subzoning (RCC-RCC, etc.). The details of

activating subzone overlays are described in the ACCEPT Geometry section.

Non-conformal subzoning affects the implementation of the electron trapping logic. With non-conformal subzones it is not sufficient to determine that an electron is trapped within a subzone; it is further necessary to determine whether the electron is trapped within the zone, if it is possible that the subzone lies on the zone boundary.

There are currently three methods for imposing subzone overlays. The first is to explicitly state the overlay to be used for subzoning the zone. The second method, which applies only to CG zones, is to intersect the first code zone with the simple body (or bodies) in such a way that the first one body (or two bodies, for a shell) in the description of the input zone specifies the entity to be used for subzoning. The additional intersection should result in a logically equivalent description of the input zone, since the original input zone description is required to be completely enclosed by that with which it is intersected (but this remains the user's responsibility). The subzone structure is now based on the first body (or first two bodies, for a shell). In both of the first two methods, it is the user's responsibility to choose a shape that makes sense; i.e., the subzone boundaries should describe a reasonable profile through the zone of interest. The third and most automated method, which applies only to CAD zones, is to base the subzoning on the RPP that is the zone bounding box.

The specified subzoning structure may describe some subzones that are completely outside of the original input zone. It is desirable not to include output for such subzones, and an attempt is made to exclude them from both the output file and the finite-element-format output written to the fort.3 file.

For CG, two checks can be made to determine whether to exclude a subzone from output. These checks are made if the "GEOMETRY 4" option is used. It is performed by comparing a point inside the subzone with the zone. (This point is chosen as the simple average of the eight vertices of the subzone. The vertices are subzone "corner" points that are used if finite-element formatted output is requested.) If the point is not within the zone, then the subzone is flagged for exclusion from the finite-element output.(For example, if the original zone is a quarter bend of a torus and the subzone structure is based on subzoning the full torus with the number of angular subzones around the circumference of the bend divisible by four, then only the subzones corresponding to the quarter bend of the torus will be included in the output.) If the full subzoning is not conformal, but all individual subzones are entirely inside or entirely outside of the zone, then this will be sufficient to accurately calculate subzone volumes.

The second CG check can provide a more rigorous determination of whether subzones coincide with the zone. For nonconformal subzoning to use accurate volume information, the volumes (of the portion of subzones within the zone) must be supplied. Volumes can be input via the "GEOMETRY 4" option. Volumes can be calculated via a Monte Carlo calculation. The preliminary flag from the first check will be overwritten if the subzone has a non-zero volume. This covers the case where part of the subzone (containing the interior point) lies outside the input zone. It should not be possible to find the interior point inside the zone for a subzone with non-zero volume overlapping the zone. The code flags this condition as an error.

For CAD, one of two methods can be used for determining whether to exclude a subzone from output. For the "GEOMETRY 3" option, the volume is calculated for the intersection of each subzone with the zone. For the "GEOMETRY 4" option, the volumes of subzones can be input directly. (It may be useful to use the former option once and subsequently use the latter option to input the calculated volumes directly to avoid the repeated computational expensive of the volume intersections.) In either case, the output will be suppressed for subzones with zero volume.

Last Modified: January 9, 2004

19 Random Number Generators

Monte Carlo calculations rely on the use of pseudo-random numbers to simulate the stochastic nature of physical processes. Therefore, one must be careful to correctly implement the available random number generator (RNG). We provide the user a choice of 3 RNGs. In this document, we first discuss the properties of the RNGs provided and then share the issues of concern we have encountered in the implementation and use of RNGs.

19.1 Portable Random Number Generators

RNGs have been made available as compile options in ITS to relieve the user of concerns about the integrity and implementation of the intrinsic RNG on his system. All 3 RNGs are double precision and as implemented, never generate zeros or ones.

RNG1, an implementation of a RNG in the MCNP code[36], has a cycle length of 2^{46} and allows easy access to its seed. However, RNG1 is not implemented as a parallel RNG. RNG2 and RNG3 will function in both serial and parallel. To permit a common interface for the RNGs, RNG1 is used to seed each batch performed with RNG2 or RNG3. Regardless of which RNG is selected in the code, the user will be seeding and receiving seed information from RNG1. Because the states of RNG2 and RNG3 are quite large, they will be written to a file when required for debugging. Only the RESTART-HISTORY feature requires access to these states.

RNG2 is an implementation of the RANMAR RNG[37, 38]. It has a state space of 2^{4656}[39], consisting of numerous cycles with an average cycle length of approximately 2^{100}[38]. RNG1 is used to randomly seed RNG2 within its state space. RNG2 requires a single seed between 0 and 900,000,000 inclusively. The first 1000 (parameter MAXCHK in randat.h) seeds from RNG1 are checked to ensure that batches in a run with RNG2 are not initialized with the same seed.

RNG3 is an implementation of the Mersenne Twister RNG[40]. It has a cycle length of 2^{19937}-1. Each batch in a run with RNG3 is seeded with a combination of 624 random numbers from RNG1.

19.2 Range

The random numbers that are generated should be uniformly distributed between 0.0 and 1.0 exclusively. There are certain places in the Monte Carlo software where random numbers that are identically 0.0 or 1.0 are unacceptable, resulting in fatal execution errors.

19.3 Access to the Seed

The state of a RNG consists of those variables that are used to determine the next random number in the sequence. The state of RNG1 consists of a single variable. Access to this variable is required to assure independent random number sequences in separate runs, to have a restart capability, and to facilitate debugging. If an error condition is detected while the Monte Carlo is in progress, for RNG1 the starting random-number seed for the current source particle (variable RIRSAV in randat.h) is printed within the terminating subroutine, ABORTX. For RNG2 and RNG3, the starting random-number seed for the current batch and the number of random-numbers generated in the batch before the current history (variables RIRA and CNRN(2) in randat.h) are printed, and the initial RNG state for the current source particle is printed to a file.

One symptom of incorrect implementation of a RNG may be the obtaining of "0" percent statistical uncertainties for all quantities; this may happen if the same seed is being used to start each batch so that the results from each batch are identical.

19.4 Reproducibility

A RNG is designed to produce the same sequence of random numbers for the same starting seed.

19.5 Cycle Length

Computer generated random numbers have a finite cycle length (the number of random numbers generated before the cycle repeats itself). One should never run so many source particles in a given run so as to exceed the RNG cycle length. A single source particle will likely use many random numbers. The number of random numbers used in a calculation is presented in the output.

19.6 Speed

Timings indicate that for our implementations RNG2 is generally faster than RNG3 and that both RNG2 and RNG3 are faster than RNG1. However, these timings vary depending on the platform and compiler.

Last Modified: January 9, 2004

20 Adjoint Calculations

The adjoint calculation mode is a complement to the forward calculation mode. In some situations where forward calculations are inefficient, adjoint calculations offer an efficient alternative. In general, if one is interested in a variety of responses with distributions in space, angle, and energy (e.g., dose and charge deposition distributions) due to a limited number of radiation sources (e.g., a monoenergetic electron beam), then it may be more efficient to use the forward method. However, if one is interested in a limited number of responses (e.g., dose at a point) due to a large number of radiation sources with distributions in space, angle, and energy (e.g., sources from various directions), then it may be more efficient to use the adjoint method.

From linear algebra, given an inner product (,), an operator T, and two functions S and R, the following equality holds:

$$(TS, R) = (S, T^{\dagger}R),$$

where $T^{\dagger}$ is the adjoint operator of T. For our purposes in the MITS code, we can think of the T as the forward mode of the computer program, and we can think of $T^{\dagger}$ as the adjoint mode of the computer program. Running MITS in both forward and adjoint modes to calculate a single response R due to a single source S should yield identical results (within statistics) since the code is solving the same problem using the same cross sections only with different methods.

The forward computer program T "operates" on a single forward source S, which must be described in space, energy, and angle. For Monte Carlo, we can think of the program T as tracking particles, and the particle source is specified by S. The result of TS is the particle flux. If the user wants a value of dose, for example, that is the value of the inner product, then the flux must be folded with the appropriate response R to obtain that value. For dose, the response is the restricted stopping power for the region of interest. For a single source, the particle flux TS can be folded with many different responses to obtain, for example, doses in various materials throughout the geometry, dose profiles within a single material, charge deposition profiles, escaping particle distributions as a function of energy and/or angle, etc. As a convenience to the user, our codes will compute the inner product for a library of common responses.

Similarly, the adjoint computer program $T^{\dagger}$ operates on a single response R, so the user must specify what single type of response (e.g., dose) is desired. For Monte Carlo, we can think of the program $T^{\dagger}$ as tracking "particles" called adjunctons, and the "adjuncton source" corresponds to the response R. The result of $T^{\dagger}R$ is a particle importance map, that gives the importance of particles (as a function of species type, space, energy, and angle) to the specified response. The inner product can now be evaluated for many different types of forward sources S. As a convenience to the user, our codes will compute the inner product for a library of common sources.

In forward mode, the distribution (in space, energy, and angle) and the species type of a single source must be specified using the POSITION, ENERGY or SPECTRUM, DIRECTION, and ELECTRON, PHOTON, or NEUTRON keywords. Then, a variety of responses may be selected. Energy and charge deposition are calculated throughout the problem by default. The user may request escape distributions (using the ELECTRON-ESCAPE, PHOTON-ESCAPE, NEUTRON-ESCAPE, and ESCAPE-SURFACES keywords) and flux distributions (using the ELECTRON-FLUX, PHOTON-FLUX, and NEUTRON-FLUX keywords).

In adjoint mode, a single response must be specified with the DETECTOR-RESPONSE keyword. The user must specify the type of response as DOSE, CHARGE, KERMA, or ESCAPE and must describe some properties (such as spatial extent) of the detector. This specifies the space, energy, and angle distribution of an adjuncton source. Then, a variety of sources may be selected. The user must request at least one source, since there are no defaults. The user may specify a surface source of each

species (ELECTRON-SURFACE-SOURCE, PHOTON-SURFACE-SOURCE, and NEUTRON-SURFACE-SOURCE) and a volume source of each species (ELECTRON-VOLUME-SOURCE, PHOTON-VOLUME-SOURCE, and NEUTRON-VOLUME-SOURCE).

In addition to the choice of spatial distributions of sources in adjoint mode, the user has the option of "binning" in energy and angle. This binning represents dividing the energy and angle domains into separate, independent sources of energy and angle extents given by the bin. Thus, if one divides a surface source into 9 equal polar-angle bins, then the output will contain the requested response for a source with a distribution from 0 to 10 degrees, a source from 10 to 20 degrees, etc. Such data may be post-processed assigning weighted distributions in angle and energy to determine the response to a wide variety of combinations of energy and angular source distributions. On the other hand, it may be more accurate (due to the approximation of binning the sources before folding with the actual distributions) and the output will be greatly condensed if the user is able to specify an angular distribution of interest (such as COSINE-LAW), the energy spectra of interest (using the SPECTRUM keyword), and allow the code to collapse the energy and angle binning structure. The choice between folding with source distributions during the calculation or in post-processing is left to the user.

Last Modified: February 2, 2004

21 Regression Testing of ITS

This document lists the steps involved in performing regression testing for a CVS commit, covers each step in detail, and discusses regression test coverage.

21.1 Regression Testing Steps

1. Store a backup copy of your changes (in case files become tainted while testing).
2. Make sure that you know what you are testing to commit. Consider using the following.
   (a) "make distclean" and "cvs diff Makefile.in" so that executables will be built properly.
   (b) "cvs diff -D now" to ensure that changes are what you want to commit. Differences may be due to changes in the repository (see step 2d and repeat).
   (c) Do a clean checkout, "cvs checkout -D now -d its-clean its", and diff it with your changes (e.g., "diff -r -b its-changes its-clean"). This will reveal added or deleted files.
   (d) "cvs update -A" will remove tags and update files to the current repository.
3. Checkout the tests, if necessary, using "cvs checkout its/Tests/RegTests".
4. Position cross sections for tantalum/aluminum from XGEN and CEPXS.
5. Run tests in serial using "regtests.pl". Evaluate results and store, if necessary.
   Warning: Running regression tests deletes the Diff and NewOutput directories!
6. Run tests in parallel using "regtests.pl -mpi". For platforms on which mpi calculations can not be run directly:
   (a) "regtests.pl -mpipre"
   (b) Execute each test. (itsjanus.com or itscrater.com may be used to run these tests.)
   (c) "regtests.pl -mpipost". Evaluate results and store, if necessary.
7. Run "regtests.pl -check" if regression outputs have changed substantially.
8. "interrupt_cleanup" to remove unwanted files from regression testing.
9. Examine the changes to be committed. (Refer to step 2.) Check if repository has changed.
10. Commit only files containing changes. Include brief descriptions of changes.
11. Notify other developers of changes, preferably via email.

21.2 Regression Testing Details

1. It is a good idea to store a backup copy of your changes at various stages of the testing and commit process. Errors may occur in the scripts, in the use of scripts, or in moving files around. Diffing the file structure after the testing process with a version set aside before testing is a simple check of any inadvertent changes made during testing. It can also serve as a reminder of changes that were necessary. For example, a bug fix implemented during parallel testing should be re-tested in serial before the commit.

2. Before beginning the testing process, you should make sure that you know what you are testing to commit. You should consider examining in detail the changes you have made. This allows you to assess any inadvertent changes that may cause the tests to fail, any changes you may expect to see in the regression test results, whether additional changes ought to be made before going through the commit process, and whether any functionality of the code may be diminished or damaged. Consider using the following techniques for examining changes:

   (a) The tests will not be able to successfully build executables if make, object, and executable files are left in place. "make distclean" can be used to remove these files. "cvs diff Makefile.in" can be used to determine if changes may prevent scripts from properly selecting code definitions.

   (b) "cvs diff -D now" can be used to evaluate how files currently existing in the repository have been changed. You should consider that differences may be the result of changes in repository files, in which case it may be necessary to update the code (see step 2d) and then diff your files again. An overview of the files changed can be obtained by grepping the output of the cvs diff for "RCS file".

   (c) Another technique for comparing altered code to the repository is to do a clean checkout, "cvs checkout -D now its", and diff your changes with the current checkout (e.g., "diff -r -b its-commit its-clean"). This, unlike a cvs diff, will reveal any added or deleted files. If a file has been inadvertently deleted, an update can be used to restore the file (e.g., "cvs update Makefile.in"). Added files (that only appear in the altered version) may be construction debris that needs to be removed. Overviews of changed files or of added and deleted files can be obtained by grepping the output of the diff for "diff -r -b" or "Only in", respectively.

   (d) Any checkout tags must be removed before files can be committed. "cvs update -A" will remove tags and update the files to the current repository. This may or may not be needed. Tags, if present, will be located in a Tag file in the CVS subdirectory of each directory checked out of the repository.

3. The test scripts should be run from within the directory structure of the altered code. If the Tests were not originally checked out with the code, it will be necessary to add the directories to the file structure. From below the its directory (i.e., from within the its directory do a "cd .."), you should checkout the tests using "cvs checkout its/Tests/RegTests".

4. There are five tantalum/aluminum cross section files necessary to perform the regression tests. Two files from XGEN are necessary. They must be located in the SHOME/cross3 directory and must be named "taala" and "taalp". Three files from CEPXS are necessary. They must be located in the SHOME/crossm directory and must be named "taal.11", "taalncp.11", and "taalp.11". These cross section files must correspond to the versions of XGEN and CEPXS that were used to generate the current regression test results in the repository. Running the XGEN and CEPXS regression tests will create these files and properly position them. However, if these regression tests have some variation

in their results (such as changes in the last digit of cross sections) due to being run on a new platform, then they may cause the ITS regression tests to fail. In such cases, the user should use the cross section files that are contained in the XGEN and CEPXS repositories.

5. There are two sets of serial tests to be executed, one containing combinatorial geometry (CG) models and the other containing CAD models. Both sets of tests will be executed by "regtests.pl" within the RegTests directory. Several flags may be used with the script to limit the number of tests performed or to control the execution of the tests. Only the CG tests will be performed if the -cg flag is used. Only the CAD tests will be performed if the -cad flag is used. By default the CAD tests will use the ACIS 6.0 libraries, but the -acis4 flag will cause the tests to use the ACIS 4.0 libraries. The -noopt flag will perform the tests without compiler optimization.

   (a) When the regression tests have completed, the user must evaluate the results. If there were changes in the output files not deemed to be test failures, store the output files by moving the files from NewOutput to Output.

   Each regression test will delete the Diff and NewOutput directories before beginning its tests, so it is important to move files from the NewOutput directory that one wishes to save.

   There are a number of ways in which tests can succeed or fail. The simplest indication that the tests have succeeded is that files (one for each test performed) located in the RegTests/Diff directory are all of size zero. The simplest indication that tests are failing will be the appearance of ohoh files in the RegTests directory. Ohoh files may contain sufficient information to determine the cause of the failure. If they do not, then files in the relevant $HOME/tmp directory should be examined.

   A more complicated scenario (that does not necessarily indicate whether the tests have passed or failed) is the presence of files that are not of size zero in the RegTests/Diff directory. In this case, the diff files should be examined to determine whether the differences from the ITS output files constitute a failure of the regression tests. Even if the diff files are not of size zero, it is possible that the regression tests have passed, if the changes in the output were expected and/or are an explainable, acceptable side effect of the code changes. If the changes revealed in the diff files are not explainable and acceptable, the code must be fixed and the regression tests must be rerun before a commit can be performed.

   WARNING: If the tests have failed without having completed all of the tests, the "interrupt_cleanup" script should be run. If this is not performed, code modifications that are necessary for some tests may remain in place and cause other tests to fail.

6. There are two sets of parallel tests to be executed, as with the serial tests. The same flags are available in parallel as in serial, but there are additional flags for specifying the parallel testing. The -mpi flag requests that parallel tests be compiled, run, and post-processed. If executed as "regtests.pl -mpi", all tests should run to completion, and the results can then be evaluated. For functionality on the crater platform, the scripts are hardwired to move the tests to "/chome/bin/tmp" and execute the tests with "mpirun -np X", where X is the number of processors needed for the test. On some platforms, the parallel regression tests must be compiled, run, and post-processed as separate steps. In that case, the -mpipre flag specifies that the tests are to be compiled. The its*com scripts can be used (submitted from below the tmp directory) to run the calculations. The -mpipost flag specifies that the tests are to be post-processed.

   (a) To compile the tests, the command "regtests.pl -mpipre" should be used.

(b) "itscrater.com" is an example script for executing the CG tests on a platform without a queuing system. "itsjanus.com" is an example script to be used with a queuing system. The files "itscratercad.com" and "itsjanuscad.com" can similarly be used for the CAD tests. The following instructions are specific to the ASCI Red platform.

Scratch space is available on janus under /scratch/tmp_(number), where (number) can be 1 to 12. Users should set up their own subdirectory (e.g., /scratch/tmp_8/$USERNAME). This scratch space will be referred to here as $SCRATCH. All calculations performed on janus should be performed on scratch space for improved execution efficiency and to minimize communication between janus and sasn100 that impacts other users.

The entire directory structure of $HOME/tmp should be copied to $SCRATCH (e.g., cp -r $HOME/tmp $SCRATCH). The $HOME/tmp directory should then be renamed. After the tests have been successfully executed, the directory may be deleted. A script, RegTests/itsjanus.com, can be used to execute all of the tests and also should be copied to $SCRATCH. The job is submitted to janus using the qsub command as "qsub -eo -re -ro -q snl.day -lP 5 -lT 600 itsjanus.com". A copy of this command is available on the first comment line of the .com file. When the calculation has completed, some large files that are not necessary for post-processing should be deleted to minimize communication back to sasn100. From the $SCRATCH/tmp directory, execute "rm */fort.* */its.x". Then, copy the $SCRATCH/tmp directory back to $HOME/tmp. After the post-processing has been successfully completed, the $SCRATCH/tmp directory may be deleted.

(c) The tests are then post-processed from the RegTests directory using "regtests.pl -mpipost".

As with the serial regression tests, when the post-processing has been completed, the user must evaluate the results. If there were changes in the output files not deemed to be test failures, the user should store the output files by moving the files from NewOutput to OutputMPI.

Each parallel post-processing regression test script will delete the Diff and NewOutput directories before beginning to process outputs, so it is important to move files from the NewOutput directory that one wishes to save.

WARNING: If the tests have failed without having completed all of the tests, the "interrupt_cleanup" script should be run. If this is not performed, code modifications that are necessary for some tests may remain in place and cause other tests to fail.

7. If regression test outputs have changed significantly (e.g., due to a change in physics or a change in the random number sequences used), then the user should execute "regtests.pl -check" in the RegTests directory. This script will compare the outputs in Output and OutputMPI from the same tests run in serial and parallel. The results should be the same. However, the resulting diff files in the directory RegTests/Diff will not have size zero, due to parallel processing output. File sizes in the Diff directory should be approximately 2kB. If the file sizes are much larger, the user should determine the reason for the differences in the serial and parallel results. Differences at this point will most likely be caused by message passing problems in the parallel implementation.

   WARNING: Do not proceed unless all regression tests have passed.

8. After performing the regression tests, some unnecessary files will remain (e.g., the Diff directory). To remove these files execute the script "interrupt_cleanup". This script will remove all files from the $HOME/tmp directory, remove the directories RegTests/Diff and RegTests/NewOutput. It will also attempt to perform other tasks that may have been necessary if one of the regression test scripts had been interrupted, but should not affect any files after a successfully completed script.

9. As in step 2, you should examine the file changes that you intend to commit. Refer to step 2 for tips on inspecting files. Performing a cvs diff, diffing with a backup copy of files created before testing, and diffing with a current checkout of the repository are useful steps. The latter will yield the most complete information. This is a convenient time to construct a list of all files containing changes that need to be committed and construct the brief descriptions of code changes to be included at the time of the commit. The list of files to be committed may be extracted semi-automatically using the techniques from step 2 used to obtain an overview of altered files.

   You should also check that the repository has not changed. This can be accomplished in two ways. First, one can simply perform a cvs update of the files to be committed and ensure that no files are updated. Second, a cvs diff can be performed between the current version and the version corresponding to (or prior to) when the previous update was performed. For example, if you estimate that an update was performed 4 hours ago, then you can check to see if there have been any changes during that time using: cvs diff -D "5 hours ago" -D now its. If there have been changes in the repository, then the code must be updated, and the regression tests must be performed again before doing a commit.

10. Files are committed using the command "cvs commit" followed by the pathnames and files to be committed. This command may be executed from anywhere within the checked out repository structure (e.g., from within the its directory "cvs commit Code/Ffiles/output.F" or from within the Ffiles directory "cvs commit output.F"). A brief description of the changes being committed may be inserted with a "-m" flag (e.g., cvs commit -m "Additional diagnostics information for MITS forward code." output.F). If this flag is not used, an editor window should be launched to provide the user with more space for describing the changes.

    To remove a file from the repository, first remove the file from the checked-out version of the repository (e.g., rm output.F), schedule the file to be removed from the repository (e.g., cvs remove output.F), and then commit the removal of the file (e.g., cvs commit output.F). The file will still be available when a checkout is performed of older versions of the code.

    To add a file to the repository, schedule the file to be added to the repository (e.g., cvs add output.F) and then commit the addition of the file (e.g., cvs commit output.F).

    To move a file within the repository structure, remove the file from one location, add it to another location, and commit the changes. For example, cvs remove Ffiles/output.F, cvs add Code/Ffiles/output.F, cvs commit Ffiles/output.F Code/Ffiles/output.F. Note that while old versions of the code can still be obtained, the new file will not "know" of its relationship to the old file, and for example, an update of the file containing alterations cannot be performed across this transition. To facilitate updating, the repository should be tagged before and after such a commit and the commit should only represent moving the file – not changing the file. The tags should be communicated to other users.

11. The final step in a commit is notifying the other developers of the changes that were made. This is most important so that others know when they must perform an update. However, it should also serve as a means for communicating any known issues resulting from the commit and provide the opportunity for review of code changes.

21.3 Regression Test Coverage

The regression test problems currently included in the test suite are discussed in this section. Serial regression tests are listed in Table 8. Parallel regression tests are listed in Table 9. The names of the tests are directly related to file names in the repository. Input files lack the number that designates a random number generator, since the same input file can be used with any random number generator. The input files also include a ".inp" suffix. Output files include a ".out" suffix.

Table 8. Serial regression test problems and preprocessor definitions

Name	Code	RNG	Other Options	Description
itsacc1	ACCEPT	RNG1		Brems convertor
itsacc2	ACCEPT	RNG2		Brems convertor
itsacch1	ACCEPT	RNG1		Pulse height
itsacch3	ACCEPT	RNG3		Pulse height
itsaccm3	ACCEPT	RNG3	MCODES	Thick vacuum gap with field
itsaccp2	ACCEPT	RNG2	PCODES	Brems convertor
itscadm2	CAD	RNG2		Mirroring, Photon-only
itscyl3	CYLTRAN	RNG3		Brems convertor
itscylm11	CYLTRAN	RNG1	MCODES	Thin vacuum gap with field
itscylm12	CYLTRAN	RNG2	MCODES	Thin vacuum gap with field
itscylm23	CYLTRAN	RNG3	MCODES	Thick vacuum gap with field
itscylp2	CYLTRAN	RNG2	PCODES	Brems convertor
itstig2	TIGER	RNG2		Brems convertor
itstigp1	TIGER	RNG1	PCODES	Brems convertor
itstigp3	TIGER	RNG3	PCODES	Brems convertor
itstigph3	TIGER	RNG3	PCODES	Pulse-height
mitsacc1	ACCEPT	RNG1	MITS	Brems convertor
mitsacc2	ACCEPT	RNG2	MITS	Brems convertor
mitsaccc2	ACCEPT	RNG2	MITS	Adjoint volume charge
mitsaccd3	ACCEPT	RNG3	MITS	Adjoint point dose
mitsacce2	ACCEPT	RNG2	MITS	Adjoint escape electrons
mitsacck3	ACCEPT	RNG3	MITS	Adjoint volume kerma
mitsaccp3	ACCEPT	RNG3	MITS	Line radiation
mitscadd2	CAD	RNG2	MITS	Point dose, direction-sphere
mitscadk2	CAD	RNG2	MITS	Point kerma, direction-sphere
mitscadm2	CAD	RNG2	MITS	Mirroring
mitscadray	CAD	RNG2	MITS	Point kerma, raytrace
mitscadsh2	CAD	RNG2	MITS	Shell subzoning
mitscadsz2	CAD	RNG2	MITS	Single body subzoning
mitstig3	TIGER	RNG3	MITS	Brems convertor
mitstigc3	TIGER	RNG3	MITS	Adjoint point charge
mitstigd1	TIGER	RNG1	MITS	Adjoint volume dose
mitstigd2	TIGER	RNG2	MITS	Adjoint volume dose
mitstige3	TIGER	RNG3	MITS	Adjoint escape electrons
mitstigk2	TIGER	RNG2	MITS	Adjoint volume kerma

Table 9. Parallel regression test problems and preprocessor definitions (all use MPI)

Name	Code	RNG	Other Options	Description
itsacc2	ACCEPT	RNG2	DYNAMIC	Brems convertor
itsacch3	ACCEPT	RNG3		Pulse height
itsaccm3	ACCEPT	RNG3	MCODES	Thick vacuum gap with field
itsaccp2	ACCEPT	RNG2	PCODES, DYNAMIC	Brems convertor
itscadm2	CAD	RNG2		Mirroring, Photon-only
itscyl3	CYLTRAN	RNG3		Brems convertor
itscylm12	CYLTRAN	RNG2	MCODES, DYNAMIC	Thin vacuum gap with field
itscylm23	CYLTRAN	RNG3	MCODES	Thick vacuum gap with field
itscylp2	CYLTRAN	RNG2	PCODES, DYNAMIC	Brems convertor
itstig2	TIGER	RNG2	DYNAMIC	Brems convertor
itstigp3	TIGER	RNG3	PCODES	Brems convertor
itstigph3	TIGER	RNG3	PCODES	Pulse-height
mitsacc2	ACCEPT	RNG2	MITS	Brems convertor
mitsaccc2	ACCEPT	RNG2	MITS	Adjoint volume charge
mitsaccd3	ACCEPT	RNG3	MITS, DYNAMIC	Adjoint point dose
mitsacce2	ACCEPT	RNG2	MITS	Adjoint escape electrons
mitsacck3	ACCEPT	RNG3	MITS, DYNAMIC	Adjoint volume kerma
mitsaccp3	ACCEPT	RNG3	MITS, DYNAMIC	Line radiation
mitscadd2	CAD	RNG2	MITS	Point dose, direction-sphere
mitscadk2	CAD	RNG2	MITS	Point kerma, direction-sphere
mitscadm2	CAD	RNG2	MITS	Mirroring
mitscadray	CAD	RNG2	MITS	Point kerma, raytrace
mitscadsh	CAD	RNG2	MITS	Shell subzoning
mitscadsz	CAD	RNG2	MITS	Single body subzoning
mitstig3	TIGER	RNG3	MITS, DYNAMIC	Brems convertor
mitstigc3	TIGER	RNG3	MITS, DYNAMIC	Adjoint point charge
mitstigd2	TIGER	RNG2	MITS	Adjoint volume dose
mitstige3	TIGER	RNG3	MITS, DYNAMIC	Adjoint escape electrons
mitstigk2	TIGER	RNG2	MITS	Adjoint volume kerma

Most of the regression tests in forward mode are based on a similar problem: a 1 MeV electron beam is incident on a slab (or large-diameter disk) of tantalum. Behind the slab of tantalum is a gap (not explicitly modeled in TIGER calculations) and then a slab of aluminum. All possible output tallies are activated (e.g., electron-flux, photon-escape, etc.). Since this problem is based on a bremsstrahlung convertor, bremsstrahlung scaling (or electron-to-photon scaling in MITS) is activated, except in problems testing the pulse-height capability since scaling is not allowed with that tally. Some deviations from this general problem for forward calculations are the ITS CAD calculation that has a photon source and photon-only transport (since continuous-energy electrons cannot be transported in CAD yet). For tests involving the MCODES, a magnetic field is active in the vacuum between the two disks and pulls electrons from the tantalum to the aluminum.

There are two regression tests in forward mode that are based upon a Monte Carlo volume calculation method. These two CAD tests are set up to test the subzone volume calculation logic, and if run long enough would verify the volumes (as void fluxes) via Monte Carlo tallies.

Most of the regression tests in adjoint mode are loosely based upon the bremsstrahlung convertor problem. Each of the detector-responses is tested for both ACCEPT and TIGER. Whereas detector tallies are activated in forward calculations, source tallies are activated in adjoint calculations. Therefore, both electron-surface-source and photon-surface-source tallies are activated.

Three adjoint regression tests are not based on the convertor problem. In these adjoint calculations, a point detector-response (of either dose or kerma) is determined within a block of tantalum. These calculations use photon-surface-source tallies with the direction-sphere logic.

The calculations performed in the regression test suite were chosen to try to maximize the coverage of preprocessor directive combinations with the fewest possible runs. The coverage of preprocessor combinations only attempts to cover the more important definitions and does not consider combinations of 3 or more definitions that may be used simultaneously. For example, the PCODES definition is tested with each random number generator; the PCODES definition is tested with ACCEPT, CYLTRAN, and TIGER; and each random number generator is tested with ACCEPT, CYLTRAN, and TIGER; but the PCODES definition is not tested for each combination of random number generator and ACCEPT, CYLTRAN, and TIGER.

A reduced set of identical tests are performed in parallel, as in serial. The set is reduced because RNG1 is not functional for parallel calculations. The results from parallel and serial calculations of these tests should be identical, except for parallel processing information.

Last Modified: January 9, 2004

A Running XGEN

This document contains outlines for running the XGEN code. Instructions are provided for running the code by using commands or by using scripts. Known platform dependencies of the code are discussed in the last section.

A.1 Running XGEN without Scripts

CHECKOUT: Do a "cvs checkout xgen" to acquire a copy of the code on crater or scorpio. If necessary, tar the directory and transfer it to the desired platform.

To run on any platform, either use the sendn script (described in the Scripts section) or build an executable using the following commands.

MAKEFILE SETTINGS: In the directory xgen/Code, you may alter the Makefile.in settings in the "scripted settings" section to specify the PCODES definition for your build of the ITS executable. The default is to build XGEN for production of the standard cross sections.

CONFIGURE: Execute "./configure". If the platform and operating system are identified, but the proper config/mh-* and config/mt-* files are not present, then you must either identify the proper files to use or create the necessary files and alter the configure.in file accordingly. If the platform and operating system are not identified, the config.sub file also must be altered (search for tflops as an example).

MAKE: If configure functions properly, a working Makefile will be produced. Execute this with the "make" command to produce the XGEN executable, xgen.x.

ATOMIC DATA: The code xgen.x will look for the atomic data file in the local directory in the Fortran unit 9 file. This file is in the cvs directory under xgen/Code/XSdata. A soft link may be created from the location of the file to one's working directory as:

ln -s $HOME/xgen/Code/XSdata/x6.dat fort.9

REGRESSION TESTS: Files for running regression tests are in xgen/Tests/RegTests. The tests can be run by executing "regtests.pl". The script will place cross section files in $HOME/cross3, output and cross section files in the directory NewOutput, and results of diffs with the repository version of output and cross section files in the directory Diffs.

A.2 Running XGEN with Scripts

Crater and Scorpio have direct access to the CVS repository. On all other platforms, the directory of files must be tarred and transferred, and the directory options in the sendn script must always be used instead of the CVS options.

For working on platforms that do not have direct access to the repository, it is recommended that two copies of the repository be set up: one copy to be used as a working directory in which code modifications and builds can be performed, and one unaltered copy that can be compared with to maintain a record of changes made to the working version. The unaltered copy should be give a name uniquely indicating the cvs version. For example, if the check out was performed as "cvs checkout -D February 1, 2002 xgen", then the unaltered copy might be named "xgen01Feb2002". The name of the directory will be the indication of the version, which is a very important key to repeating a calculation at some later time.

1. DOCUMENTATION and SCRIPTS: Use the command "cvs checkout xgen" to acquire a copy of the code and all associated files.

   (a) Documentation is available in the xgen/Docs directory. These are LaTeXfiles that must be compiled.

   (b) From the directory xgen/Scripts, copy sendn and the nxCVS script to your $HOME/bin directory. Copy the files in the Subscripts directory to a $HOME/bin/Subscripts directory.

2. CUI FILE: Copy the xgen/Scripts/x.cui file to your working directory, and edit it for your specific problem. The sections of a cui file are:

   (a) DRIVER SCRIPT: The script nxCVS is available as a driver script. You may substitute a customized script by including it in the first section of the cui file. Sendn will position the cui file, including the sections of the driver script. The driver script contains the commands necessary to build and execute the program, clean up after itself, and produce relevant result information in a job file. (If necessary, it will also include in the job file information useful in determining the cause of a job failure).

   (b) DEFS: The PCODES preprocessor definition may be selected in the defs section of the cui file.

   There are several options for running the scripts that may be set in the defs section of the cui file. Compiler flags may be specified. The user may request email notification that the scripts have finished.

   The user may also request an interactive script. The interactive script should be used on systems that require job queuing or on systems that require cross compiling. The first half of the script will build the executable. The files for running the job will be located in $HOME/tmp/-<jobname>. The executable is named "xgen.x", the input file is "xdat", and the output file should be named "output". When the calculation is complete, the user may execute the "postproc" file in the $HOME/tmp/<jobname> directory. This will move the fort.11 cross section file to $HOME/cross3 directory, rename it according to the response that was given to the sendn script, and produce a job file in the directory from which the job was originally submitted.

   (c) DIFFS: Patches to be applied to the code should be inserted in the "diffs" portion of the xgen.cui file. These patches can be formatted as a cvs diff or as a directory diff. Diffs can be lifted out of job files from previous calculations. Alternatively, one can checkout a copy of the code, make modifications, and then generate a diffs file. On a platform with access to the CVS repository, one can use "cvs diff > diffs" from within the xgen/Code directory. On a platform without access to the CVS repository one can perform a directory diff, but this must be performed from within the xgen/Code directory of the unaltered copy of the code, and it must use the -r and -b flags (in that order), such as "diff -r -b . $HOME/xgenaltered/Code> diffs". Then, the diffs file can be used in the xgen.cui file. Multiple diff files (diffs from two different executions of the diff command) cannot be patched to a single cvs file.

   New files can be added to a build. Within the diffs section, a line of the following format denotes the start of a new file:

   New file: <Directory/Filename>

   To be included in the compiling and linking of the code, the new file must be referenced in the Makefile.in list of sources. This change in the Makefile.in can also be included in the diffs section by making the desired change in a copy of Makefile.in and using the cvs diff procedure described above.

(d) XDAT: The XGEN input should be included in the last section of the cui file. Instructions for XGEN input are in Keywords for XGEN.

3. SENDN: Submit the job using the command "sendn x.cui <jobname>". This script will prompt you for the following 3 pieces of information (and possibly the 4th depending upon your responses to the first 3):

   (a) CROSS SECTIONS: First, sendn will request the name of the cross section file <cross> to be created. Upon successful completion, the scripts will assign the given name to the fort.11 cross section file and place it at "$HOME/cross3/<cross>".

   (b) LOCAL COMPILE: Second, sendn will request the location of a checked-out copy of xgen that can be used for making the executable. Thus, you may use a version of the code that you have checked out and modified. You must give a complete pathname for the base xgen directory (e.g., /scratch/temporary/xgen). No files will be deleted from the make directory as a result of the calculation, but some files may be modified if requested in the diffs section of the cui file. Any new files that have been included in the directory (other than through the diffs section of the cui file) will not appear in the job file, but they may be used if the Makefile.in has been so modified (in which case, the job file will show the reference to the new file as a difference in the Makefile.in).

   Be aware that any definitions specified in the "defs" section of the cui file and any modifications specified in the "diffs" section of the cui file will be applied to the files in this directory. Attempts to change the code definitions for a calculation using the same directory will not take effect unless the Makefile.in has been reset and "make clean" has been executed. (sendn will copy Makefile.in to Makefile.in.defs if such a file does not already exist. This can be used to reset the Makefile.in.) Attempts to apply the diffs in a directory where the diffs have already been applied for a previous calculation will result in an error.

   CVS COMPILE: To request a cvs checkout of the code, you may respond to this request with "none" (or press "Enter"). The cvs checkout will be performed in the $HOME/tmp/<jobname> directory and should not affect any other versions of the code.

   (c) DIRECTORY DIFF for LOCAL COMPILE: If your response to the second request was a directory pathname, the script will request a local directory with which to perform a diff. The diff command will be used to compare the two directories and all subdirectories. The job file will not record the version number of either directory, therefore the user may not have enough information in the job file to duplicate a calculation unless the diff directory has a name corresponding to the tag used in the cvs checkout of the code. The directory name does appear in the job file.

   VERSION for CVS COMPILE: If your response to the second request was "none", the script will request the version for a cvs checkout. The command will be issued as "cvs checkout <options> xgen". The response to this request will be used for the <options> and any valid cvs flags may be used that do not contain a slash, "/". Examples of valid syntax for responses are: -D now, -D "March 28, 2001", -D 4:00pm, -D "3 hours ago", -D "2 fortnights ago", -r 1, -r release-1.0, etc. Another valid response is to simply press "Enter" with no input, which will result in the checkout of the most recent version of the code.

   (d) CVS DIFF for LOCAL COMPILE: If you gave a pathname for making the executable but not for a directory diff, the script will request the version for a cvs diff. The syntax for responses to this request are the same as for specifying the cvs version for a checkout. The directory in which the executable is made will be compared with the repository using the cvs diff command. This

version (that will appear in the job file) and the results of the cvs diff (that will also appear in the job file) can be used to reproduce a calculation.

CVS UPDATE/DIFF for CVS COMPILE: If you did not give a pathname for making the executable, the script will request a version for a cvs update and diff. In this case, the script will check out a version of the code using the options in the third response, apply the "diffs" from the cui file, attempt to update to the version of the code specified in this response, and then compare the resulting code to the repository using the cvs diff command with the options specified in this response. The version requested here (that will appear in the job file) and the results of the cvs diff (that will also appear in the job file) can be used to reproduce a calculation.

For either of these options, if no option is specified for the cvs diff, the option "-D now" will be used. The date and time of the calculation, which are recorded in the job file, and the results of the cvs diff can be used to reproduce the calculation at a later date.

A.3 Platform Dependencies

The following is a list of platforms on which XGEN has been successfully built and tested. Following the name of the platform is the system type and operating system.

- Scorpio (IBM, AIX4)
- Virgo (IBM, AIX4)
- Crater (AMD Athlon, Linux)
- Gollum (DEC Alpha, OSF1 V4)

Last Modified: January 9, 2004

B XGEN Code Options

Only one code option can be selected for compiling the XGEN codes. This option has been implemented as a preprocessor definition. It is necessary to choose between the standard cross section code and the PCODES version.

The code option must be specified in either the CUI file (if using the scripts to execute the code) or in the "scripted settings" section of the Makefile (if building an executable). Refer to the documentation on Running XGEN for more details on applying definitions.

B.1 Preprocessor Definitions

PCODES (more ionization and relaxation)

Last Modified: April 16, 2004

C Summary of XGEN Keywords

Table C.1 contains a listing of keywords relevant to the XGEN code. The keywords are listed approximately in an order of importance. In addition, for each keyword the default code behavior is listed. The default behavior will be employed by the code if the keyword is not found in the input deck. More detailed descriptions of the syntax, subkeywords, and use of these keywords are contained in the XGEN Keywords section.

Table C.1. Summary of XGEN keywords and default settings

KEYWORD	DEFAULT
ENERGY	1.0 MeV
MATERIAL	AL
element symbol(s)	required
CONDUCTOR or NON-CONDUCTOR	see discussion under keywords
GAS	normal state (pure elements only) liquid/solid (compounds/mixture)
DENSITY	normal density (g/cc) (pure elements only)
DENSITY-RATIO	1.0
SUBSTEP	internal
TITLE	no title
ELECTRON-GRID-LENGTH	off
STEP	8
PRINT-ALL	abbreviated cross-section tables are printed
ECHO	off

Last Modified: April 16, 2004

D Keywords for XGEN

The input keywords must be specified in either the CUI file (if using the scripts to execute the code) or in the input file (if executing the code manually). Refer to the documentation on Running XGEN for more details on specifying the input file in the CUI file or otherwise. An overview of the keywords is available in the Summary of XGEN Keywords section.

D.1 Input Notation

Most keywords should be used once and not repeated in an input file. Exceptions to this are the ECHO and MATERIAL keywords. Most subkeywords should be used once per use of their primary keyword.

Parameters are associated with the preceding keyword appearing on the same line. If parameters are omitted, they will be set to zero. Consideration should be given to the fact that in some situations this value is invalid and will trigger an error.

Comments may be inserted in the input deck. Anything appearing to the right of an asterisk anywhere in the input deck will be treated as a comment and ignored by the code.

Input is not case sensitive, with one insignificant exception. Character input provided with the TITLE keyword will appear in the output file exactly as provided, but the case will not affect the code in any way.

D.2 Keywords

1. **ECHO**

   Syntax: ECHO [parameter(1)]

   Example: ECHO 1

   Default: no echo

   If "ECHO 1" is inserted in the input stream, all subsequent input card images will be echoed to the Fortran unit 6 output.

2. **ELECTRON-GRID-LENGTH**

   Syntax: ELECTRON-GRID-LENGTH [parameter(1)]

   Example: ELECTRON-GRID-LENGTH 80

   Default: 64, which corresponds to 8 halvings (or 0.39%) of the maximum energy. Regardless of the value, the grid will be truncated below 1 keV.

   In problems requiring a broad energy range of electron transport, this keyword allows the user to extend the energy range for which electron transport data is available. Alternatively, if the STEP keyword is used to refine the electron energy-loss grid, this keyword can be used to preserve the energy range over which data is available.

3. **ENERGY**

   Syntax: ENERGY [parameter(1)]

   Example: ENERGY 2.5

Default: Maximum cross-section energy is 1.0 MeV

Maximum energy in MeV for which electron cross sections will be calculated.

4. **MATERIAL**

Syntax: MATERIAL [parameter(1)] [parameter(2)] ...

Example: MATERIAL TA 0.25 C 0.75

Default: Aluminum

Identifies unique material (pure element, compound or homogeneous mixture) and the appropriate weight fractions (defaults to 1.0 for pure elements) for which electron and photon cross sections are to be calculated. This keyword is repeated for each unique material. DATA arrays containing 100 atomic symbols (e.g., TA for tantalum) along with corresponding default values for the electrical characterization (conductor/non-conductor), mass density, and state (solid/liquid or gas) at normal pressure and temperature (zero °C and one atm) are included in the code to simplify the input for pure materials. The default properties of elements are provided in Table D.2. To override these defaults or to construct compound materials the following secondary keywords associated with this primary keyword may be used.

(a) **element symbol(s)**

Syntax: element symbol [parameter(1)]

Example: TA 0.25 Al 0.75

Default: no default

Except for pure elements where a blank for parameter(1) will default to 1.0, each element symbol must be followed by a single real number, parameter(1), which is the weight fraction of that constituent. The weight fraction must sum to 1.0.

(b) **CONDUCTOR/NON-CONDUCTOR**

Syntax: NON-CONDUCTOR

Example: NON-CONDUCTOR

Default: A pure element with a Z of 1, 2, 7, 8, 9, 10, 17, 18, 35, 36, 53, 54, 85, or 86 is a non-conductor; otherwise, the element is a conductor. A compound/mixture is a non-conductor if any one of its constituent elements is a non-conductor by default; otherwise, it is a conductor.

The only collective effect in the ITS Monte Carlo model is the density-effect correction to the electronic stopping power. The value of this correction depends on whether the transport region is a conductor or non-conductor. The user may explicitly define any material to be a conductor or a non-conductor via the appropriate keyword. However, if a material so defined as a conductor consists of constituent elements, all of which are non-conductors by default (e.g., pure water), the material will be redefined to be a non-conductor, and the user will be so informed via a message in the output file.

(c) **GAS**

Syntax: GAS

Example: GAS

Default: Normal state for elements and liquid/solid for compounds

This keyword is used to specify that this material is in a gaseous state at normal pressure and temperature. The material state is used in calculating the density effect contribution to the electronic stopping power.

(d) **DENSITY**

Syntax: DENSITY [parameter(1)]

Example: DENSITY 2.0

Default: Normal density for elements – no default for compounds!

Density of the target material at normal pressure and temperature (g/cm3).

(e) **DENSITY-RATIO**

Syntax: DENSITY-RATIO [parameter(1)]

Example: DENSITY-RATIO 0.5

Default: Density ratio is 1.0

Ratio of the actual density to the density of the target material at normal pressure and temperature (used in calculating density effect contribution to electronic stopping powers).

(f) **SUBSTEP**

Syntax: SUBSTEP [parameter(1)]

Example: SUBSTEP 10

Default: Calculated internally as a function of atomic number

Number of random walk substeps into which each macroscopic electron step is subdivided. The default values have been empirically determined; other values should not be used without careful consideration of their effects on the condensed history model.

5. **PRINT-ALL**

Syntax: PRINT-ALL

Example: PRINT-ALL

Default: Abbreviated cross-section tables will be printed

This keyword will cause all except differential electron cross-sections to be printed out.

6. **STEP**

Syntax: STEP [parameter(1)]

Example: STEP 12

Default: Successive electron energies are related by $E_{i+1}=2^{-1/8}E_i$

Parameter that determines spacing of electron energy grid and the size of the macroscopic electron steps. Successive energies are related by $E_{i+1}=2^{-(1/[parameter(1)])}E_i$. The default value has been empirically determined; other values should not be used without careful consideration of their effects on the condensed history model.

7. **TITLE**

Syntax: TITLE

Example: TITLE

Adjoint Dose calculation in Al box in Satellite GPS-4

Default: no title

This keyword signals that the next line of input contains a character string that is a title of up to 80 columns that will be written to the output file.

Table D.2. List of available elements and default properties in XGEN

Z	Element	Symbol	Atomic Weight	Density (g/cm^3)	State	Conductor
1	Hydrogen	H	1.008	0.08988E-3	Gas	N
2	Helium	He	4.0026	0.1785E-3	Gas	N
3	Lithium	Li	6.94	0.53	S/L	Y
4	Beryllium	Be	9.01218	1.85	S/L	Y
5	Boron	B	10.81	2.34	S/L	Y
6	Carbon	C	12.011	2.26	S/L	Y
7	Nitrogen	N	14.0067	1.2506E-3	Gas	N
8	Oxygen	O	15.9994	1.429E-3	Gas	N
9	Fluorine	F	18.9984	1.696E-3	Gas	N
10	Neon	Ne	20.17	0.89990E-3	Gas	N
11	Sodium	Na	22.9898	0.97	S/L	Y
12	Magnesium	Mg	24.305	1.74	S/L	Y
13	Aluminum	Al	26.9815	2.70	S/L	Y
14	Silicon	Si	28.086	2.33	S/L	Y
15	Phosphorus	P	30.9738	1.82	S/L	Y
16	Sulfur	S	32.06	2.07	S/L	Y
17	Chlorine	Cl	35.453	3.214E-3	Gas	N
18	Argon	Ar	39.948	1.7837E-3	Gas	N
19	Potassium	K	39.102	0.86	S/L	Y
20	Calcium	Ca	40.08	1.55	S/L	Y
21	Scandium	Sc	44.9559	3.00	S/L	Y
22	Titanium	Ti	47.90	4.51	S/L	Y
23	Vanadium	V	50.941	6.10	S/L	Y
24	Chromium	Cr	51.996	7.19	S/L	Y
25	Manganese	Mn	54.938	7.43	S/L	Y
26	Iron	Fe	55.847	7.86	S/L	Y
27	Cobalt	Co	58.9332	8.90	S/L	Y
28	Nickel	Ni	58.71	8.90	S/L	Y
29	Copper	Cu	63.546	8.96	S/L	Y
30	Zinc	Zn	65.37	7.14	S/L	Y
31	Gallium	Ga	69.72	5.91	S/L	Y
32	Germanium	Ge	72.59	5.32	S/L	Y
33	Arsenic	As	74.9216	5.72	S/L	Y
34	Selenium	Se	78.96	4.79	S/L	Y
35	Bromine	Br	79.904	7.59E-3	Gas	N
36	Krypton	Kr	83.80	3.733E-3	Gas	N
37	Rubidium	Rb	85.467	1.53	S/L	Y
38	Strontium	Sr	87.62	2.60	S/L	Y
39	Yttrium	Y	88.9059	4.47	S/L	Y
40	Zirconium	Zr	91.22	6.49	S/L	Y

Table D.2 (continued).

Z	Element	Symbol	Atomic Weight	Density (g/cm^3)	State	Conductor
41	Niobium	Nb	92.9064	8.40	S/L	Y
42	Molybdenum	Mo	95.94	10.2	S/L	Y
43	Technetium	Tc	98.9062	11.5	S/L	Y
44	Ruthenium	Ru	101.07	12.2	S/L	Y
45	Rhodium	Rh	102.9055	12.4	S/L	Y
46	Palladium	Pd	106.4	12.0	S/L	Y
47	Silver	Ag	107.868	10.5	S/L	Y
48	Cadmium	Cd	112.4	8.65	S/L	Y
49	Indium	In	114.82	7.31	S/L	Y
50	Tin	Sn	118.69	7.30	S/L	Y
51	Antimony	Sb	121.75	6.62	S/L	Y
52	Tellurium	Te	127.6	6.24	S/L	Y
53	Iodine	I	126.9045	4.94	S/L	N
54	Xenon	Xe	131.3	5.887E-3	Gas	N
55	Cesium	Cs	132.9055	1.90	S/L	Y
56	Barium	Ba	137.34	3.50	S/L	Y
57	Lanthanum	La	138.9055	6.17	S/L	Y
58	Cerium	Ce	140.12	6.67	S/L	Y
59	Praeseodymium	Pr	140.9077	6.77	S/L	Y
60	Neodymium	Nd	144.24	7.00	S/L	Y
61	Promethium	Pm	145.0	7.22	S/L	Y
62	Samarium	Sm	150.4	7.54	S/L	Y
63	Europium	Eu	151.96	5.26	S/L	Y
64	Gadolinium	Gd	157.25	7.89	S/L	Y
65	Terbium	Tb	158.9254	8.27	S/L	Y
66	Dysprosium	Dy	162.50	8.54	S/L	Y
67	Holmium	Ho	164.9303	8.80	S/L	Y
68	Erbium	Er	167.26	9.05	S/L	Y
69	Thulium	Tm	168.9342	9.33	S/L	Y
70	Ytterbium	Yb	173.04	6.98	S/L	Y
71	Lutetium	Lu	174.97	9.84	S/L	Y
72	Hafnium	Hf	178.49	13.1	S/L	Y
73	Tantalum	Ta	180.947	16.6	S/L	Y
74	Tungsten	W	183.85	19.3	S/L	Y
75	Rhenium	Re	186.2	21.0	S/L	Y
76	Osmium	Os	190.2	22.6	S/L	Y
77	Iridium	Ir	192.22	22.5	S/L	Y
78	Platinum	Pt	195.09	21.4	S/L	Y
79	Gold	Au	196.9665	19.3	S/L	Y
80	Mercury	Hg	200.59	13.6	S/L	Y

Table D.2 (continued).

Z	Element	Symbol	Atomic Weight	Density (g/cm^3)	State	Conductor
81	Thallium	Tl	204.37	11.85	S/L	Y
82	Lead	Pb	207.2	11.4	S/L	Y
83	Bismuth	Bi	208.9806	9.8	S/L	Y
84	Polonium	Po	209.0	9.2	S/L	Y
85	Astatine	At	210.0	No default	S/L	N
86	Radon	Rn	222.0	9.73E-3	Gas	N
87	Francium	Fr	223.0	No default	S/L	Y
88	Radium	Ra	226.0	5.00	S/L	Y
89	Actinium	Ac	227.0	10.07	S/L	Y
90	Thorium	Th	232.0381	11.7	S/L	Y
91	Protactinium	Pa	231.0359	15.4	S/L	Y
92	Uranium	U	238.029	19.07	S/L	Y
93	Neptunium	Np	237.0482	19.5	S/L	Y
94	Plutonium	Pu	244.0	19.84	S/L	Y
95	Americium	Am	243.0	11.7	S/L	Y
96	Curium	Cm	247.0	13.51	S/L	Y
97	Berkelium	Bk	247.0	14.0	S/L	Y
98	Californium	Cf	251.0	No default	S/L	Y
99	Einsteinium	Es	254.0	No default	S/L	Y
100	Fermium	Fm	257.0	No default	S/L	Y

Last Modified: January 9, 2004

E Running CEPXS

This document contains outlines for running the CEPXS code. Instructions are provided for running the code by using commands or by using scripts. Known platform dependencies of the code are discussed in the last section.

E.1 Running CEPXS without Scripts

CHECKOUT: Do a "cvs checkout cepxs" to acquire a copy of the code on crater or scorpio. If necessary, tar the directory and transfer it to the desired platform.

To run on any platform, either use the sendn script (described in the Scripts section) or build an executable using the following commands.

CONFIGURE: In the directory cepxs/Code, execute "./configure". If the platform and operating system are identified, but the proper config/mh-* and config/mt-* files are not present, then you must either identify the proper files to use or create the necessary files and alter the configure.in file accordingly. If the platform and operating system are not identified, the config.sub file also must be altered (search for tflops as an example).

MAKE: If configure functions properly, a working Makefile will be produced. Execute this with the "make" command to produce the CEPXS executable, xcepxs.

ATOMIC DATA: The code xcepxs will look for the atomic data files in the local directory. These files are in the cvs directory under cepxs/Code/XSdata. Soft links may be created from the location of the atomic data files to one's working directory as:

ln -s $HOME/cepxs/Code/XSdata/elec3.dat elec3.dat

REGRESSION TESTS: Files for running regression tests are in cepxs/Tests/RegTests. The "runregtests" script will make an executable, run each problem, place output files in the directory NewOutput, and place results of diffs with the repository version of output files in the directory Diffs.

E.2 Running CEPXS with Scripts

Crater and Scorpio have direct access to the CVS repository. On all other platforms, the directory of files must be tarred and transferred, and the directory options in the sendn script must always be used instead of the CVS options.

For working on platforms that do not have direct access to the repository, it is recommended that two copies of the repository be set up: one copy to be used as a working directory in which code modifications and builds can be performed, and one unaltered copy that can be compared with to maintain a record of changes made to the working version. The unaltered copy should be give a name uniquely indicating the cvs version. For example, if the check out was performed as "cvs checkout -D February 1, 2002 cepxs", then the unaltered copy might be named "cepxs01Feb2002". The name of the directory will be the indication of the version, which is a very important key to repeating a calculation at some later time.

1. DOCUMENTATION and SCRIPTS: Use the command "cvs checkout cepxs" to acquire a copy of the code and all associated files.

(a) Documentation is available in the cepxs/Docs directory. These are LaTeX files that must be compiled.

(b) From the directory cepxs/Scripts, copy sendn and the ncCVS script to your $HOME/bin directory. Copy the files in the Subscripts directory to a $HOME/bin/Subscripts directory.

2. CUI FILE: Copy the cepxs/Scripts/cepxs.cui file to your working directory, and edit it for your specific problem. The sections of a cui file are:

(a) DRIVER SCRIPT: The script ncCVS is available as a driver script. You may substitute a customized script by including it in the first section of the cui file. Sendn will position the cui file, including the sections of the driver script. The driver script contains the commands necessary to build and execute the program, clean up after itself, and produce relevant result information in a job file. (If necessary, it will also include in the job file information useful in determining the cause of a job failure).

(b) DEFS: There are currently no preprocessor definitions to be set for CEPXS.

There are several options for running the scripts that may be set in the defs section of the cui file. Compiler flags may be specified. The user may request email notification that the scripts have finished.

The user may also request an interactive script. The interactive script should be used on systems that require job queuing or on systems that require cross compiling. The first half of the script will build the executable. The files for running the job will be located in SHOME/tmp/<jobname>. The executable is named "xcepxs". When the calculation is complete, the user may execute the "postproc" file in the SHOME/tmp/<jobname> directory. This will move the fort.11 cross section file to SHOME/crossm directory, rename it according to the response that was given to the sendn script, and produce a job file in the directory from which the job was originally submitted.

(c) DIFFS: Patches to be applied to the code should be inserted in the "diffs" portion of the cepxs.cui file. These patches can be formatted as a cvs diff or as a directory diff. Diffs can be lifted out of job files from previous calculations. Alternatively, one can checkout a copy of the code, make modifications, and then generate a diffs file. On a platform with access to the CVS repository, one can use "cvs diff > diffs" from within the cepxs/Code directory. On a platform without access to the CVS repository one can perform a directory diff, but this must be performed from within the cepxs/Code directory of the unaltered copy of the code, and it must use the -r and -b flags (in that order), such as "diff -r -b . SHOME/cepxsaltered/Code> diffs". Then, the diffs file can be used in the cepxs.cui file. Multiple diff files (diffs from two different executions of the diff command) cannot be patched to a single cvs file.

New files can be added to a build. Within the diffs section, a line of the following format denotes the start of a new file:

New file: <Directory/Filename>

To be included in the compiling and linking of the code, the new file must be referenced in the Makefile.in list of sources. This change in the Makefile.in can also be included in the diffs section by making the desired change in a copy of Makefile.in and using the cvs diff procedure described above.

(d) CEPINP: The CEPXS input should be included in the last section of the cui file. Instructions for CEPXS input are in cepxs/Docs/keyCEPXS.tex.

3. SENDN: Submit the job using the command "sendn cepxs.cui <jobname>". This script will prompt you for the following 3 pieces of information (and possibly the 4th depending upon your responses to the first 3):

(a) CROSS SECTIONS: First, sendn will request the name of the cross section file <cross> to be created. Upon successful completion, the scripts will assign the given name to the fort.11 cross section file and place it at "SHOME/crossm/<cross>.11".

(b) LOCAL COMPILE: Second, sendn will request the location of a checked-out copy of cepxs that can be used for making the executable. Thus, you may use a version of the code that you have checked out and modified. You must give a complete pathname for the base cepxs directory (e.g., /scratch/temporary/cepxs). No files will be deleted from the make directory as a result of the calculation, but some files may be modified if requested in the diffs section of the cui file. Any new files that have been included in the directory (other than through the diffs section of the cui file) will not appear in the job file, but they may be used if the Makefile.in has been so modified (in which case, the job file will show the reference to the new file as a difference in the Makefile.in).

Be aware that any definitions specified in the "defs" section of the cui file and any modifications specified in the "diffs" section of the cui file will be applied to the files in this directory. Attempts to change the code definitions for a calculation using the same directory will not take effect unless the Makefile.in has been reset and "make clean" has been executed. (sendn will copy Makefile.in to Makefile.in.defs if such a file does not already exist. This can be used to reset the Makefile.in.) Attempts to apply the diffs in a directory where the diffs have already been applied for a previous calculation will result in an error.

CVS COMPILE: To request a cvs checkout of the code, you may respond to this request with "none" (or press "Enter"). The cvs checkout will be performed in the $HOME/tmp/<jobname> directory and should not affect any other versions of the code.

(c) DIRECTORY DIFF for LOCAL COMPILE: If your response to the second request was a directory pathname, the script will request a local directory with which to perform a diff. This should be the unaltered copy that you made when you checked out. The diff command will be used to compare the two directories and all subdirectories. The job file will not record the version number of either directory, therefore the user may not have enough information in the job file to duplicate a calculation unless the diff directory has a name corresponding to the tag used in the cvs checkout of the code. The directory name does appear in the job file.

VERSION for CVS COMPILE: If your response to the second request was "none", the script will request the version for a cvs checkout. The command will be issued as "cvs checkout <options> cepxs". The response to this request will be used for the <options> and any valid cvs flags may be used that do not contain a slash, "/". Examples of valid syntax for responses are: -D now, -D "March 28, 2001", -D 4:00pm, -D "3 hours ago", -D "2 fortnights ago", -r 1, -r release-1.0, etc. Another valid response is to simply press "Enter" with no input, which will result in the checkout of the most recent version of the code.

(d) CVS DIFF for LOCAL COMPILE: If you gave a pathname for making the executable but not for a directory diff, the script will request the version for a cvs diff. The syntax for responses to this request are the same as for specifying the cvs version for a checkout. The directory in which the executable is made will be compared with the repository using the cvs diff command. This version (that will appear in the job file) and the results of the cvs diff (that will also appear in the job file) can be used to reproduce a calculation.

CVS UPDATE/DIFF for CVS COMPILE: If you did not give a pathname for making the executable, the script will request a version for a cvs update and diff. In this case, the script will check out a version of the code using the options in the third response, apply the "diffs" from the cui file, attempt to update to the version of the code specified in this response, and then compare the resulting code to the repository using the cvs diff command with the options specified in this

response. The version requested here (that will appear in the job file) and the results of the cvs diff (that will also appear in the job file) can be used to reproduce a calculation.

For either of these options, if no option is specified for the cvs diff, the option "-D now" will be used. The date and time of the calculation, which are recorded in the job file, and the results of the cvs diff can be used to reproduce the calculation at a later date.

E.3 Platform Dependencies

The following is a list of platforms on which CEPXS has been successfully built and tested. Following the name of the platform is the system type and operating system.

- Scorpio (IBM, AIX4)

  This machine uses architecture definition AIX to include exception handling.

- Virgo (IBM, AIX4)

  This machine uses architecture definition AIX to include exception handling.

- Taos (Sun, Solaris 5.7)

  Regression test results were generated on Taos.

- Luigi (Sun, Solaris 5.8)

- Linux (x86, Redhat 7.X)

- Crater (AMD Athlon, Linux)

- Janus (i386, TFLOPS), compile on sasn100 (Sun, Solaris 5.6)

  The configure command must specify the target:

  ```
  configure -target=i386-tflops
  ```

  Janus uses architecture definition TFLOPS to use double precision instead of extended precision.

- Gollum (DEC Alpha, OSF1 V4)

- PC

  No configuration script, Makefile, or regression test scripts are available. It may be necessary to use double precision instead of extended precision.

  There has been limited testing on this platform.

Last Modified: January 9, 2004

F Summary of CEPXS Keywords

Table F.3 contains a listing of keywords for the CEPXS code for generating cross sections to be used with the MITS code. The keywords are listed approximately in order of importance. In addition, for each keyword the default code behavior is listed. The default behavior will be employed by the code if the keyword is not found in the input deck. More detailed descriptions of the syntax, subkeywords, and use of these keywords are contained in the CEPXS Keywords section.

Table F.3. Summary of CEPXS keywords (for use with MITS) and default settings

KEYWORD	DEFAULT
MITS	ONELD
**** MATERIALS ****	
MATERIAL or MATNAM	required (repeated for each material)
**** ENERGY RANGE ****	
ENERGY	1.0 MeV
CUTOFF	1 keV for photon sources 1% of source energy for electron sources
**** SPECIES/COUPLING ****	
ELECTRON-SOURCE or PHOTON-SOURCE	electron source, full-coupling with photons,
or NEUTRON-SOURCE	and no neutrons
NO-POSITRONS	positrons automatic if energy extends above 1.03 MeV and full-coupling
**** GROUP STRUCTURE ****	
ELECTRONS (or EGROUP)	50 logarithmic groups
PHOTONS (or PGROUP)	50 logarithmic groups
NEUTRONS	no default structure
ANNIHILATION-LINE	automatic line if energy extends above 1.03 MeV
NO-LINES	annihilation line is automatic
LINES	relaxation lines are mixed with continuum
MONO-PHOTON	no source line groups
**** ALTERNATIVE PHYSICS MODELS ****	
ITS2P1	cross sections correspond to ITS 3.0
USCAT and/or USCAT-ITS	Fokker-Planck scattering cosine = 0.95
NO-PCODE	ITS-PCODES ionization/relaxation physics
NO-SEC-ELEC-GLOBAL	secondary electrons generated
NO-COHERENT	coherent photon scattering included
NO-INCOH-BINDING	incoherent scattering includes binding effects
CSDA	restricted CSDA
KNOCKONS-WITH-PRIMARIES	secondaries w/o corresp. primary downscatter
LEGENDRE	Legendre order is 15
ELASTIC-LEGENDRE	Legendre order is 15
**** OTHER OPTIONS ****	
TITLE	no title
PRINT	cross section matrices not printed
PRINT-ALL	no "additional" information is printed
CONTRAST	compact fort.11 file
CEPXS-INFO-ONLY	cross sections are generated

Last Modified: February 2, 2004

G Keywords for CEPXS

The input keywords must be specified in either the CUI file (if using the scripts to execute the code) or in the input file (if executing the code manually). Refer to the documentation on Running CEPXS for more details on specifying the input file in the CUI file or otherwise. An overview of the keywords is available in the Summary of CEPXS Keywords section.

G.1 Input Notation

Most keywords should be used once and not repeated in an input file. Exceptions to this are the MATERIAL, MATNAM, and MONO-PHOTON keywords. Most subkeywords should be used once per use of their primary keyword.

Once a keyword has been used, then parameters in brackets are optional and parameters not in brackets are required.

Comments may be inserted in the input deck. Lines with an asterisk in the first column will be ignored by the code. It is important to note that asterisks inserted anywhere other than the first column may not cause words that follow to be ignored by the code.

Input is not case sensitive, with one significant exception. File names given under the ELEMENTS subkeyword of the NEUTRON-SOURCE keyword will be read as case-sensitive character strings. Also, character input provided with the TITLE keyword will appear in the output file exactly as provided, but the case does not affect the code in any way.

G.2 Keywords

Transport Code Keywords

1. **ONELD**

   Syntax: ONELD
   Example: ONELD
   Default: ONELD

   This keyword specifies that the cross sections from CEPXS will be formatted for ONELD.

2. **MITS**

   Syntax: MITS
   Example: MITS
   Default: ONELD

   This keyword specifies that the cross sections from CEPXS will be formatted for MITS.

3. **BFP** *(Not MITS, Not ONELD)*

   Syntax: BFP
   Example: BFP

Default: The Boltzmann-Fokker-Planck approximation is not used.

This keyword specifies that the Boltzmann-Fokker-Planck approximation is to be used for electron cross sections. This approximation is automatically used if the MITS keyword is used.

Additional Keywords

1. **ANNIHILATION-LINE**

   Syntax: ANNIHILATION-LINE

   Example: ANNIHILATION-LINE

   Default: Annihilation line is included if energy grid extends above 1.03 MeV, unless keyword NO-LINES is used. (See the LINES keyword for explanation of line groups.)

   Specifies that the annihilation line will be separated from the continuum, if 0.511 MeV is within the energy range. This keyword overrides the NO-LINES keyword and overrides the requirement that the energy grid extend above 1.03 MeV.

   Note: This keyword is incompatible with PGROUP.

2. **BCD** *(NOT CURRENTLY FUNCTIONAL)*

   Syntax: BCD

   Example: BCD

   This keyword specifies the format needed for MCNP.

3. **CEPXS-INFO-ONLY**

   Syntax: CEPXS-INFO-ONLY

   Example: CEPXS-INFO-ONLY

   Default: Cross sections are calculated.

   This keyword specifies that only range, mfp, and other such information is to be generated, but cross sections will not be calculated.

4. **CONTRAST** *(MITS)*

   Syntax: CONTRAST

   Example: CONTRAST

   Default: A compact fort.11 file is produced for MITS. (No fort.ll file is produced unless MITS is specified.)

   An expanded fort.11 file is produced that can be compared with fort.11 files from other versions of CEPXS using the contrast program. This fort.11 file is still functional for MITS, but it has larger memory requirements.

5. **CONDENSED-HISTORY** *(Not MITS)*

Syntax: CONDENSED-HISTORY

Example: CONDENSED-HISTORY

Default: A diamond energy-differencing scheme is used for the CSDA operator.

A second order backward differencing of the CSDA operator is used, and a condensed-history based approximation is used.

6. **CSDA**

Syntax: CSDA

Example: CSDA

Default: Restricted CSDA is used.

The Continuous-Slowing-Down Approximation (CSDA) is used to characterize all electron energy loss.

7. **CSDLD** *(Not MITS)*

Syntax: CSDLD

Example: CSDLD

Default: A diamond energy-differencing scheme is used for the CSDA operator.

8. **CUTOFF**

Syntax: CUTOFF [parameter(1)]

Example: CUTOFF 0.01

Default: 1 keV for photon sources. One percent of the source energy for electron sources.

Cross sections extend to the cutoff energy. This energy is the lower bound of the lowest energy group for both electrons and photons. [parameter(1)] is the cutoff energy in MeV, i.e., it specifies the lower energy bound of the lowest-energy group to be generated. The cutoff energy cannot be less than 1 keV.

9. **EGROUP**

Syntax: EGROUP

Example: EGROUP

Default: Fifty logarithmic electron groups.

The electron group structure is obtained from the ASCII file, "egroup". An "egroup" file is generated on each CEPXS run and may be used on a subsequent CEPXS run.

The structure of the "egroup" file is not important unless the user wishes to create this file from scratch. On the first line of the file, the number of electron groups is specified. Each subsequent line of the file is associated with a group (in descending order of energy.) Each line specifies, in order, the top energy of the group, the mid-point energy of the group, and the bottom energy of the group, all in MeV. The code will function provided that the mid-point energies are within the energy group bounds, even if they are not exact mid-point values.

10. **ELASTIC-LEGENDRE**

Syntax: ELASTIC-LEGENDRE [parameter(1)]

Example: ELASTIC-LEGENDRE 5

Default: 15, or Legendre order of cross sections set with LEGENDRE keyword.

This keyword specifies the Legendre order of electron elastic cross sections as [parameter(1)]. This order is used if between 0 and the Legendre order of the cross sections. Otherwise, it is reset to the Legendre order of the cross sections.

Note: This keyword should be used with caution. For ONELD the extended transport correction is applied, and it is important to consider the quadrature set with which the cross sections will be used.

11. **ELECTRONS**

Syntax: ELECTRONS

Example: ELECTRONS

Default: For MITS, 50 logarithmic electron groups. For other codes, up to 50 automatic-structure electron groups.

If this primary keyword is used, one of the secondary keywords must be used to specify the electron group structure.

Note: This keyword is incompatible with EGROUP.

(a) **LINEAR**

Syntax: LINEAR [parameter(1)]

Example: LINEAR 40

A linear electron group structure is created where [parameter(1)] is the number of electron groups.

(b) **LOG**

Syntax: LOG [parameter(1)]

Example: LOG 40

A logarithmic electron group structure is created where [parameter(1)] is the number of electron groups.

(c) **USER**

Syntax: USER [parameter(1)]
[parameter(2)] ... [parameter(parameter(1)+1)]

Example: USER 10
1.0 0.8 0.6 0.5 0.4 0.35 0.3 0.28 0.26 0.25

A user-defined electron group structure is created where [parameter(1)] is the number of electron groups. The string that follows is composed of the lower boundary energies of each group arranged in decreasing order. The last energy must be the same as the cutoff energy, i.e., [parameter(parameter(1)+1)] must equal CUTOFF [parameter(1)].

12. **ELECTRON-SOURCE**

Syntax: ELECTRON-SOURCE

Example: ELECTRON-SOURCE

Default: For MITS, a fully-coupled electron source. A source must be specified for other codes.

The following sub-keywords are used to specify the coupling scheme for photon and electron cross sections.

(a) **NO-COUPLING**

Syntax: NO-COUPLING

Example: NO-COUPLING

Default: **Full-coupling.**

This keyword indicates that photons will not be included in the calculation.

(b) **PARTIAL-COUPLING**

Syntax: PARTIAL-COUPLING

Example: PARTIAL-COUPLING

Default: Full-coupling.

This keyword indicates that electrons can produce photons but photons cannot produce electrons.

(c) **FULL-COUPLING**

Syntax: FULL-COUPLING

Example: FULL-COUPLING

Default: **Full-coupling.**

This keyword indicates that electrons can produce photons and photons can produce electrons. Also, if the upper bound of the energy grid extends above 1.03 MeV, positrons will be included in the cross sections with full coupling to photons and partial coupling to electrons.

13. **ENERGY**

Syntax: ENERGY parameter(1)

Example: ENERGY 0.665

Default: 1 MeV

parameter(1) specifies the midpoint energy in MeV of the highest-energy group to be generated (but see description of energy grid generation for caveats). This energy can not exceed 100 MeV.

14. **FIRST-ORDER** *(Not MITS)*

Syntax: FIRST-ORDER

Example: FIRST-ORDER

Default: A diamond energy-differencing scheme is used for the CSDA operator.

Cross sections associated with the CSD operator will be equivalent to a first-order differencing in energy.

15. **INCOH-BINDING**

Syntax: INCOH-BINDING

Example: INCOH-BINDING

Default: No incoherent binding for ONELD. Incoherent binding for MITS.

This keyword specifies that incoherent binding effects will be included. Either of the keywords ITS2P1 or NO-INCOH-BINDING specifies that incoherent binding effects will not be included.

16. **ITS2P1**

Syntax: ITS2P1

Example: ITS2P1

Default: ITS version 3.0 physics.

Cross sections corresponding to ITS version 2.1 are used.

17. **KNOCKONS-WITH-PRIMARIES**

Syntax: KNOCKONS-WITH-PRIMARIES

Example: KNOCKONS-WITH-PRIMARIES

Default: Knockon electrons are generated without corresponding primary downscatter.

Knockon electron production is associated with primary downscatter.

18. **LEGENDRE**

Syntax: LEGENDRE parameter(1)

Example: LEGENDRE 7

Default: The Legendre order of the cross sections is 15.

parameter(1) is the Legendre order of the cross sections. If parameter(1) is zero or omitted, the default will be used.

19. **LINES**

Syntax: LINES parameter(1) parameter(2) ...

Example: LINES AL BE

Default: photon relaxation lines are mixed with the continuum.

Specifies that separate groups are generated for the photon relaxation radiation of the elements specified by the parameters. The binding energies of these elements will also fall on the photon group boundaries (the group structure is adjusted accordingly.) If no parameters are specified, relaxation line groups will be created for all elements in the problem.

Line groups have are assigned energy group widths corresponding to the continuum group with which they would otherwise be mixed and mid-point energies equal to the line energy. The cross sections for the line groups are calculated at the line energies.

Note: This keyword is incompatible with PGROUP.

20. **MATERIAL**

Syntax: MATERIAL [parameter(1)] parameter(2) ...

Example: MATERIAL H .1111 -

O .8889

Default: None. At least one MATERIAL or MATNAM entry is required.

Specifies material composition. The mandatory string specifies either the name of a hard-wired material or the chemical names of the constituent elements and their weight fractions.

In the latter case, the first parameter in the string is mandatory and is the chemical name of an element in the material. The second parameter is the weight fraction of the element in the material. For single element materials, the weight fraction is not needed (and will be in error if not equal to 1.0).

Additional parameters may specify other elements and their weight fractions. If necessary, this parameter list may be extended to other lines by terminating a line with a dash. The weight fractions of elements in a material must sum to 1.0.

The default properties of elements are provided in Table G.4. The hardwired materials available in CEPXS and the default properties of those materials are provided in Table G.5.

(a) **CONDUCTOR**

Syntax: CONDUCTOR

Example: CONDUCTOR

Default: Default values are provided for single element materials. A compound is set as a conductor unless any one of its constituent elements is a non-conductor.

The only collective effect in the CEPXS model is the density-effect correction to the electronic stopping power. The value of this correction can depend on whether a material is a conductor or a non-conductor.

Note: If a compound is specified as a conductor, but none of its constituent elements are conductors, then the material is automatically reset to a non-conductor.

Note: This keyword is incompatible with ITS2P1.

(b) **DENSITY**

Syntax: DENSITY [parameter(1)]

Example: DENSITY 1.001

Default: For single element materials and hard-wired materials, defaults are provided. For a user-defined composition, this keyword is required.

The density of the material is set to [parameter(1)] in g/cm3.

(c) **GAS**

Syntax: GAS

Example: GAS

Default: Solid or Liquid.

This keyword specifies that the material is a gas.

(d) **NON-CONDUCTOR**

Syntax: NON-CONDUCTOR

Example: NON-CONDUCTOR

Default: Default values are provided for single element materials. A compound is set as a conductor unless any one of its constituent elements is a non-conductor.

The only collective effect in the CEPXS model is the density-effect correction to the electronic stopping power. The value of this correction can depend on whether a material is a conductor or a non-conductor.

Note: This keyword is incompatible with ITS2P1.

(e) **NO-SEC-ELEC**

Syntax: NO-SEC-ELEC

Example: NO-SEC-ELEC

Default: Secondary electrons are generated.

This keyword specifies that secondary electrons are not generated in this material.

21. **MATNAM**

Syntax: MATNAM [parameter(1)]
[parameter(2)] parameter(3) ...

Example: MATNAM WATER
H 0.1111 O 0.8889

Default: None. At least one MATERIAL or MATNAM entry is required.

This keyword is an alternative to the MATERIAL keyword. It allows the user to assign a material name to any material whether an element, a hard-wired material, or a user-defined composition. The code will use the first six characters of the user-assigned material name, [parameter(1)]. This user-assigned name will only appear in the CEPXS output file, cepout, and is not available for use in the generated cross section files.

Unlike the MATERIAL keyword, the material composition is specified on the following line. However, all of the same secondary keywords apply.

22. **MONO-PHOTON**

Syntax: MONO-PHOTON [parameter(1)]

Example: MONO-PHOTON 1.311

Default: No source line groups.

A mono-energetic source line group is created at the energy specified by [parameter(1)] in MeV. This keyword must be repeated for each source line needed. Up to 28 lines are allowed.

If the source line is meant to be the highest energy in the calculation, it should be set exactly equal to the energy specified with the ENERGY keyword. This will result in the source line being the highest energy group, with the second highest energy group spanning the energy range between it and the third highest energy group. (ENERGY specified the mid-point energy of the highest energy group. This is still the case, but now the highest energy group is a line group.) If the source line is otherwise specified to fall within the highest energy group, it will still be the first group, but the second group will extend in energy both above and below the line source group and therefore, may not yield the desired results.

If the energy specified for the source line is higher or lower than the extent of the energy grid, it will be omitted.

Note: In MITS, source lines are not allowed in adjoint mode.

Note: For ONELD, PHOTON-SOURCE must be specified to use MONO-PHOTON.

23. **NEUTRONS** *(MITS)*

Syntax: NEUTRONS

Example: NEUTRONS

Default: No neutron groups are included in the cross sections.

If this primary keyword is used, the secondary keyword USER must be used to specify the neutron group structure. The NEUTRON-SOURCE keyword must be used to specify the neutron cross sections available. This keyword must be used to specify the neutron cross sections that are to be included in the cross section file and to assign the energy group structure of those cross sections.

(a) **USER**

Syntax: USER [parameter(1)]
[parameter(2)] ... [parameter(parameter(1)+2)]

Example: USER 5
14.0 1.0 0.1 0.001 1E-5 1E-10

A user-defined neutron group structure is created where [parameter(1)] is the number of neutron groups desired in the fort.11 cross section file. Beginning on the following line is a list of numbers composed of the boundary energies of each group . The energy group boundaries must be arranged in decreasing order. Unlike the PHOTONS and ELECTRONS keywords, this list must start with the **upper** energy bound of the first energy group and **all** energy boundaries must be given, i.e., parameter(1)+1 values are expected. This group structure is independent of the ENERGY and CUTOFF keywords.

If the number of groups is less than the number read from elemental cross section data files (see NEUTRON-SOURCE/NEUTRON-GROUPS), then the first [parameter(1)] neutron group cross sections will be used. The effect of truncating the number of neutron groups upon the neutron-to-photon cross sections should be considered if photon groups are included.

If coupled neutron-photon cross sections are desired, the ENERGY and CUTOFF keywords will apply to the photon group structure (and electron group structure, if applicable). Since the neutron-to-photon cross sections have been read from a file, the photon group structure must correspond exactly to the group structure of the photon cross sections that have been read - this is a user responsibility.

If neutron-only cross sections are desired, the PHOTONS keyword must specify zero groups with "USER 0" followed by a blank line.

Note: Neutrons can only be employed if neutron cross sections are read from files specified under the NEUTRON-SOURCE keyword.

24. **NEUTRON-SOURCE** *(MITS)*

Syntax: NEUTRON-SOURCE

Example: NEUTRON-SOURCE

Default: A fully-coupled electron source.

This keyword (and the ELEMENTS sub-keyword) must be used to read neutron cross sections from user-supplied files.

Note: The NEUTRONS keyword must be used to specify the neutron energy group structure.

Note: Neutron and photon physics must always be partially coupled. The following keywords for selecting the coupling scheme refer to the photon and electron coupling physics.

(a) **NO-COUPLING**

Syntax: NO-COUPLING

Example: NO-COUPLING

Default: Full-coupling.

This sub-keyword specifies the photon to electron coupling scheme. NO-COUPLING indicates that electrons (and positrons) will not be included in the calculation. To obtain neutron-only cross sections, this option should be supplemented with by the selection of 0 photon groups using the PHOTONS keyword.

(b) **PARTIAL-COUPLING**

Syntax: PARTIAL-COUPLING

Example: PARTIAL-COUPLING

Default: Full-coupling.

This keyword specifies the photon to electron coupling scheme. PARTIAL-COUPLING indicates that photons can produce electrons but electrons cannot produce photons.

(c) **FULL-COUPLING**

Syntax: FULL-COUPLING

Example: FULL-COUPLING

Default: Full-coupling.

This keyword specifies the photon and electron physics coupling scheme. FULL-COUPLING indicates that photons can produce electrons and electrons can produce photons. Also, if the upper bound of the energy grid extends above 1.03 MeV, positrons will be included in the cross sections with coupling to photons and electrons. Note: It may be necessary to use the NO-LINES keyword to prevent the insertion of an annihilation line and preserve the photon group structure.

(d) **ELEMENTS**

Syntax: ELEMENTS [parameter(1)]

Example: ELEMENTS 2

O

Oxygen

H

Hydrog

Default: None. If the NEUTRON-SOURCE keyword is used, the elements must be specified.

This keyword indicates that cross sections will be read into CEPXS for [parameter(1)] elements. The keyword must be followed by two times [parameter(1)] lines of data. For each element, one line must contain the chemical name of the element, and the following line must contain the six character name of the file containing the elemental cross section data.

(e) **EXTERNAL-PHOTONS**

Syntax: EXTERNAL-PHOTONS

Example: EXTERNAL-PHOTONS

Default: CEPXS-generated photon cross sections are employed.

In addition to neutron-to-neutron cross sections, elemental cross section data should contain neutron-to-photon and photon-to-photon cross sections. This keyword specifies that photon-to-photon scattering cross sections that are read from the elemental cross section files are to be used.

The photon group structure must be provided by the user with the PHOTONS keyword and the USER subkeyword. If neutron-only cross sections are desired, the PHOTONS keyword must specify zero groups with "USER 0" followed by a blank line.

Note: If CEPXS-generated photon cross sections are employed (the default), the photon group structure still must correspond to the group structure of the cross sections read from elemental files. This is a user responsibility. The neutron-to-photon cross sections will not be accurate otherwise.

(f) **LEG-ORDER**

Syntax: LEG-ORDER [parameter(1)]

Example: LEG-ORDER 3

Default: Cross section data is assumed to extend to a Legendre order of 5.

This keyword specifies the Legendre order of cross section data to be read from the user-supplied files. [parameter(1)] can be less than, but not greater than, the Legendre order of cross sections available in the cross section files.

(g) **NEUTRON-GROUPS**

Syntax: NEUTRON-GROUPS [parameter(1)]

Example: NEUTRON-GROUPS 40

Default: 89 neutron groups.

This keyword specifies the number of neutron groups contained in the user-supplied cross section files. The number of groups to be processed into the fort.11 file is specified by the NEUTRONS keyword.

(h) **PHOTON-GROUPS**

Syntax: PHOTON-GROUPS [parameter(1)]

Example: PHOTON-GROUPS 50

Default: 48 photon groups.

This keyword specifies the number of photon groups contained in the user-supplied cross section files.

25. **NO-COHERENT**

Syntax: NO-COHERENT

Example: NO-COHERENT

Default: Coherent photon scattering will be included in the calculation.

This keyword deactivates the simulation of coherent photon scattering. This provides a functionality equivalent to the NO-COHERENT keyword in ITS.

26. **NO-INCOH-BINDING**

Syntax: NO-INCOH-BINDING

Example: NO-INCOH-BINDING

Default: Incoherent photon scattering will include binding effects.

This keyword causes incoherent photon scattering to be simulated in the Klein-Nishina or free-electron approximation. This provides a functionality equivalent to the NO-INCOH-BINDING keyword in ITS.

27. **NO-LINES**

Syntax: NO-LINES

Example: NO-LINES

Default: An annihilation line is included if the energy grid extends above 1.03 MeV.

This keyword specifies that no line groups will be included. The LINES keyword will override this keyword and cause photon relaxation line groups to be included. The ANNIHILATION-LINE keyword will override this keyword and cause an annihilation line group to be included.

28. **NO-PCODE**

Syntax: NO-PCODE

Example: NO-PCODE

Default: CEPXS essentially duplicates the ionization/relaxation physics of the ITS PCODES.

This keyword is used to select the ionization/relaxation physics of the standard (non-PCODES) ITS codes. This option allows only the K-shell to have non-zero binding energy. A single "average" energy for the fluorescence photons and Auger electrons is used. This "average" energy is less than the K-shell binding energy.

29. **NO-POSITRONS**

Syntax: NO-POSITRONS

Example: NO-POSITRONS

Default: Positrons are generated if the upper limit of the energy grid exceeds 1.03 MeV and cross sections are fully-coupled.

Positrons are not generated. Annihilation quanta are generated at the site of the pair interaction. The approximate treatment for positrons must be specified by one of the secondary keywords.

(a) **PEQE**

Syntax: PEQE

Example: PEQE

Default: None.

Pair secondaries are produced and transported as electrons. That is, annihilation quanta are produced at the beginning rather than end of the positron path. This option provides a reasonable estimate for dose but an incorrect charge prediction.

(b) **NO-PAIR**

Syntax: NO-PAIR

Example: NO-PAIR

Default: None.

No pair secondaries are produced. Energy and charge are deposited locally. This option provides a reasonable estimate for charge but an incorrect dose prediction.

30. **NO-SEC-ELEC-GLOBAL**

Syntax: NO-SEC-ELEC-GLOBAL

Example: NO-SEC-ELEC-GLOBAL

Default: Secondary electrons are generated in all materials.

Neither photon-produced nor electron-produced secondary electrons are generated in any material.

31. **PGROUP**

Syntax: PGROUP

Example: PGROUP

Default: Fifty logarithmic photon groups.

The photon group structure is obtained from the ASCII file, "pgroup". A "pgroup" file is generated on each CEPXS run and may be used on a subsequent CEPXS run.

The structure of the "pgroup" file is not important unless the user wishes to create this file from scratch. On the first line of the file, the number of photon groups is specified. For MITS, the first line also contains the following (in order): the value of USCAT, a logical flag indicating whether positrons are present, and a logical flag indicating whether the NO-PAIR keyword is used. Each subsequent line of the file is associated with a group (in descending order of energy.) Each line specifies, in order, the top energy of the group in MeV, the mid-point energy of the group in MeV, the bottom energy of the group in MeV, and an integer that is zero for a continuum group or one for a line group (relaxation, source, or annihilation). The code will function provided that the mid-point energies are within the energy group bounds, even if they are not exact mid-point values. For line groups, the energy bounds should be the energy bounds of the group in which the line would otherwise be included, and the mid-point energy should the line energy.

The line of "pgroup" following the energy group structure specifies the number of relaxation lines (not including annihilation and source lines) and the group that contains the annihilation line (zero if not applicable). On the following lines, one line gives the energy of a relaxation line followed by the Z value of the associated element and the next line contains three character fields (a3,a4,a4) that describe the relaxation line type.

32. **PHOTONS**

Syntax: PHOTONS

Example: PHOTONS

Default: For MITS, 50 logarithmic photon groups. For other codes, up to 50 automatic-structure photon groups.

If this primary keyword is used, one of the secondary keywords must be used to specify the photon group structure.

Note: This keyword is incompatible with PGROUP.

(a) **LINEAR**

Syntax: LINEAR [parameter(1)]

Example: LINEAR 40

A linear photon group structure is created where [parameter(1)] is the number of photon groups.

(b) **LOG**

Syntax: LOG [parameter(1)]

Example: LOG 40

A logarithmic photon group structure is created where [parameter(1)] is the number of photon groups.

(c) **USER**

Syntax: USER [parameter(1)]
[parameter(2)] ... [parameter(parameter(1)+1)]

Example: USER 10
1.0 0.8 0.6 0.5 0.4 0.35 0.3 0.28 0.26 0.25

A user-defined photon group structure is created where [parameter(1)] is the number of photon groups. The string that follows is composed of the lower boundary energies of each group arranged in decreasing order. The last energy should be the same as the cutoff energy, i.e., [parameter(parameter(1)+1)] must equal CUTOFF [parameter(1)].

33. **PHOTON-SOURCE**

Syntax: PHOTON-SOURCE

Example: PHOTON-SOURCE

Default: For MITS only, transport is independent of the source specified and the cross sections are fully-coupled by default. A source must be specified for other codes.

The following sub-keywords are used to specify the coupling scheme for photon and electron cross sections.

(a) **NO-COUPLING**

Syntax: NO-COUPLING

Example: NO-COUPLING

Default: Full-coupling.

This keyword indicates that electrons will not be included in the calculation.

(b) **PARTIAL-COUPLING**

Syntax: PARTIAL-COUPLING

Example: PARTIAL-COUPLING

Default: **Full-coupling.**

This keyword indicates that photons can produce electrons but electrons cannot produce photons.

(c) **FULL-COUPLING**

Syntax: FULL-COUPLING

Example: FULL-COUPLING

Default: Full-coupling.

This keyword indicates that photons can produce electrons and electrons can produce photons. Also, if the upper bound of the energy grid extends above 1.03 MeV, positrons will be included in the cross sections with full coupling to photons and partial coupling to electrons.

34. **PRINT**

Syntax: PRINT

Example: PRINT

Default: Multigroup Legendre cross section matrices are not printed.

This keyword specifies that the multigroup Legendre cross section matrices are to be printed to the CEPXS output file, CEPOUT.

(a) **LEG**

Syntax: LEG parameter(1)

Example: LEG 3

Default: Only the zero Legendre order cross sections are printed.

This sub-keyword specifies that multigroup Legendre cross sections are to be printed from Legendre order zero to Legendre order parameter(1).

(b) **ROWS**

Syntax: ROWS

Example: ROWS

Default: All cross section rows are printed.

This sub-keyword specifies that multigroup Legendre cross sections are to be printed for only selected rows of cross sections. More specifically, the total cross section, the energy deposition cross section, etc., will be printed, but group-to-group scattering cross sections will not be printed except for self-scatter cross sections.

35. **PRINT-ALL**

Syntax: PRINT-ALL

Example: PRINT-ALL

Default: No "additional" information will be printed.

This keyword requests that additional information be printed out for each material. This information includes:

- The collisional and radiative stopping powers and the density effect correction.
- The collisional stopping power due to large-energy losses.
- The exponent for the power-law extrapolation of the collisional stopping power below 10 keV

36. **SECOND-ORDER** *(Not MITS)*

Syntax: SECOND-ORDER

Example: SECOND-ORDER

Default: A diamond energy-differencing scheme is used for the CSDA operator.

A second order backward differencing of the CSDA operator is used.

37. **TITLE**

Syntax: TITLE parameter(1)

Example: TITLE

3 MeV electrons on gold

Default: No title.

This keyword specifies the title of the calculation that will appear in output files. The title will be read from the following parameter(1) lines. If parameter(1) is zero or omitted, the following (one) line will be read as the title. The title card may contain any alphanumeric information in columns 1 to 72 with which the user wishes to identify the calculation.

38. **USCAT** *(MITS)*

Syntax: USCAT [parameter(1)]

Example: USCAT 0.99

Default: The cosine of the Fokker-Planck scattering angle is 0.95 for all energies and materials.

This keywords sets the cosine of the Fokker-Planck scattering angle to [parameter(1)].

39. **USCAT-ITS** *(MITS)*

Syntax: USCAT-ITS

Example: USCAT-ITS

Default: The cosine of the Fokker-Planck scattering angle is 0.95 for all energies and materials.

Energy- and material-dependent values of the Fokker-Planck scattering angle are internally calculated such that the electron mean free path is equal to the electron sub-step size in ITS. However, the automatic generation of Fokker-Planck scattering angles is over-ridden in cases where the scattering cosine would be less than the constant "uscat" parameter (which has a default value of 0.95 but may be changed with the USCAT keyword).

Table G.4. List of available elements and default properties in CEPXS

Z	Element	Symbol	Atomic Weight	Density (g/cm^3)	State	Conductor
1	Hydrogen	H	1.00794	0.08375E-3	Gas	N
2	Helium	He	4.002602	0.1663E-3	Gas	N
3	Lithium	Li	6.941	0.534	S/L	Y
4	Beryllium	Be	9.012182	1.848	S/L	Y
5	Boron	B	10.811	2.37	S/L	Y
6	Carbon	C	12.0107	1.70	S/L	Y
7	Nitrogen	N	14.0067	1.165E-3	Gas	N
8	Oxygen	O	15.9994	1.332E-3	Gas	N
9	Fluorine	F	18.9984	1.580E-3	Gas	N
10	Neon	Ne	20.1797	0.8385E-3	Gas	N
11	Sodium	Na	22.98977	0.971	S/L	Y
12	Magnesium	Mg	24.305	1.74	S/L	Y
13	Aluminum	Al	26.981538	2.699	S/L	Y
14	Silicon	Si	28.0855	2.33	S/L	Y
15	Phosphorus	P	30.97376	2.2	S/L	Y
16	Sulfur	S	32.066	2.0	S/L	Y
17	Chlorine	Cl	35.4527	2.995E-3	Gas	N
18	Argon	Ar	39.948	1.662E-3	Gas	N
19	Potassium	K	39.0983	0.862	S/L	Y
20	Calcium	Ca	40.078	1.55	S/L	Y
21	Scandium	Sc	44.95591	2.989	S/L	Y
22	Titanium	Ti	47.867	4.54	S/L	Y
23	Vanadium	V	50.9415	6.11	S/L	Y
24	Chromium	Cr	51.9961	7.18	S/L	Y
25	Manganese	Mn	54.93805	7.44	S/L	Y
26	Iron	Fe	55.845	7.874	S/L	Y
27	Cobalt	Co	58.9332	8.90	S/L	Y
28	Nickel	Ni	58.6934	8.902	S/L	Y
29	Copper	Cu	63.546	8.96	S/L	Y
30	Zinc	Zn	65.39	7.133	S/L	Y
31	Gallium	Ga	69.723	5.904	S/L	Y
32	Germanium	Ge	72.61	5.323	S/L	Y
33	Arsenic	As	74.9216	5.73	S/L	Y
34	Selenium	Se	78.96	4.5	S/L	Y
35	Bromine	Br	79.904	7.072E-3	Gas	N
36	Krypton	Kr	83.80	3.478E-3	Gas	N
37	Rubidium	Rb	85.4678	1.532	S/L	Y
38	Strontium	Sr	87.62	2.54	S/L	Y
39	Yttrium	Y	88.90585	4.469	S/L	Y
40	Zirconium	Zr	91.224	6.506	S/L	Y

Table G.4 (continued).

Z	Element	Symbol	Atomic Weight	Density (g/cm^3)	State	Conductor
41	Niobium	Nb	92.90638	8.57	S/L	Y
42	Molybdenum	Mo	95.94	10.22	S/L	Y
43	Technetium	Tc	98.0	11.5	S/L	Y
44	Ruthenium	Ru	101.07	12.41	S/L	Y
45	Rhodium	Rh	102.9055	12.41	S/L	Y
46	Palladium	Pd	106.42	12.02	S/L	Y
47	Silver	Ag	107.8682	10.5	S/L	Y
48	Cadmium	Cd	112.411	8.65	S/L	Y
49	Indium	In	114.818	7.31	S/L	Y
50	Tin	Sn	118.71	7.31	S/L	Y
51	Antimony	Sb	121.76	6.691	S/L	Y
52	Tellurium	Te	127.60	6.24	S/L	Y
53	Iodine	I	126.90447	4.93	S/L	N
54	Xenon	Xe	131.29	5.485E-3	Gas	N
55	Cesium	Cs	132.90545	1.873	S/L	Y
56	Barium	Ba	137.327	3.50	S/L	Y
57	Lanthanum	La	138.9055	6.154	S/L	Y
58	Cerium	Ce	140.116	6.657	S/L	Y
59	Praeseodymium	Pr	140.90765	6.71	S/L	Y
60	Neodymium	Nd	144.24	6.90	S/L	Y
61	Promethium	Pm	145.0	7.22	S/L	Y
62	Samarium	Sm	150.36	7.46	S/L	Y
63	Europium	Eu	151.964	5.243	S/L	Y
64	Gadolinium	Gd	157.25	7.90	S/L	Y
65	Terbium	Tb	158.92534	8.229	S/L	Y
66	Dysprosium	Dy	162.50	8.55	S/L	Y
67	Holmium	Ho	164.93032	8.795	S/L	Y
68	Erbium	Er	167.26	9.066	S/L	Y
69	Thulium	Tm	168.9342	9.321	S/L	Y
70	Ytterbium	Yb	173.04	6.73	S/L	Y
71	Lutetium	Lu	174.967	9.84	S/L	Y
72	Hafnium	Hf	178.49	13.31	S/L	Y
73	Tantalum	Ta	180.9479	16.65	S/L	Y
74	Tungsten	W	183.84	19.3	S/L	Y
75	Rhenium	Re	186.207	21.02	S/L	Y
76	Osmium	Os	190.23	22.57	S/L	Y
77	Iridium	Ir	192.217	22.42	S/L	Y
78	Platinum	Pt	195.078	21.02	S/L	Y
79	Gold	Au	196.96655	19.32	S/L	Y
80	Mercury	Hg	200.59	13.55	S/L	Y

Table G.4 (continued).

Z	Element	Symbol	Atomic Weight	Density (g/cm^3)	State	Conductor
81	Thallium	Tl	204.3833	11.72	S/L	Y
82	Lead	Pb	207.2	11.35	S/L	Y
83	Bismuth	Bi	208.98038	9.747	S/L	Y
84	Polonium	Po	209.0	9.32	S/L	Y
85	Astatine	At	210.0	No default	S/L	N
86	Radon	Rn	222.0	9.066E-3	Gas	N
87	Francium	Fr	223.0	No default	S/L	Y
88	Radium	Ra	226.0	5.00	S/L	Y
89	Actinium	Ac	227.0	10.07	S/L	Y
90	Thorium	Th	232.0381	11.72	S/L	Y
91	Protactinium	Pa	231.03588	15.37	S/L	Y
92	Uranium	U	238.0289	18.95	S/L	Y
93	Neptunium	Np	237.0381	20.45	S/L	Y
94	Plutonium	Pu	244.0	19.82	S/L	Y
95	Americium	Am	243.0	13.671	S/L	Y
96	Curium	Cm	247.0	13.51	S/L	Y
97	Berkelium	Bk	247.0	14.78	S/L	Y
98	Californium	Cf	251.0	No default	S/L	Y
99	Einsteinium	Es	252.0	No default	S/L	Y
100	Fermium	Fm	257.0	No default	S/L	Y

Table G.5. List of available materials, compositions (in w/o), and densities in CEPXS

Material	Keyword	Composition	Density
Alumina	AL203	O .4708 AL .5292	3.99
Aluminum 6061	AL6061	MG .0080 AL .9725 SI .0040 TI .0015 CR .0015 MN .0015 FE .0070 CU .0015 ZN .0025	2.69
Aluminum 7075	AL7075	MG .0250 AL .9000 CR .0030 CU .0160 ZN .0560	2.80
Bakelite Phenolic	BAKELI	H .0645 C .7656 O .1699	1.45
Brass	BRASS	FE .0020 CU .6150 ZN .3520 PB .0310	8.40
Bronze	BRONZE	P .0030 CU .9470 SN .0050	8.86
CaF TLD	CAFTLD	F .4668 CA .51332 MN .01988	3.18
Kapton	KAPTON	H .0100 C .5600 N .1300 O .3000	1.42
Kevlar	KEVLAR	H .0428 C .6958 N .1166 O .1448	1.31
Kennertium	KENNER	CU .0200 W .9800	18.5
Kovar	KOVAR	MN .0030 FE .5370 CO .1700 NI .2900	7.90
Lithium Fluoride	LIF	LI .2675 F .7325	2.635
Mica	MICA	H .0051 O .4820 AL .2032 SI .2115 K .0982	2.80
Mylar	MYLAR	H .0519 C .6186 O .3295	1.38
Nylon	NYLON	H .0980 C .6368 N .1238 O .1414	1.13
Phenolic Linen	PHENOL	H .0800 C .7100 N .0600 O .1500	1.33
Polyethylene	POLYE	H .1437 C .8563	0.92
Polyimide PCB	POLYIM	H .0150 B .0120 C .2800 N .0350 O .3370 NA .0030 MG .0020 AL .0590 SI .1460 CA .1110	1.85
Polyurethane	POLYU	H .0402 C .5913 N .1398 O .2287	0.192
Silicone Dioxide	SIO2	O .5326 SI .4674	2.20
Silica Glass	GLASS	O .5257 MG .0006 AL .0048 SI .4475 K .0008 CA .0164 FE .0042	2.18
Solder	SOLDER	SN .6000 PB .4000	8.67
Stainless Steel 304	SS304	C .0008 SI .0100 P .0005 S .0003 CR .1900 MN .0200 FE .6784 NI .1000	8.03
Stainless Steel 410	SS410	C .0015 SI .0100 P .0004 S .0003 CR .1200 MN .0100 FE .8578	7.76
RTV630	RTV630	H .0490 C .2100 O .3250 AL .0010 SI .4130 CU .0020	1.28
Teflon	TEFLON	C .2401 F .7599	2.15
Water	WATER	H .1100 O .8900	1.0

Last Modified: January 9, 2004

H Glossary of Terms

ACCEPT: the three dimensional geometry capability of the ITS codes.

ACCEPTM: the MCODES version of ACCEPT.

ACCEPTP: the PCODES version of ACCEPT.

ACIS: a particular format of B-rep geometry developed by Spatial Technology, Inc.

Adjoint: the solution of the adjoint transport problem. This simulation method is similar to the physical processes going backwards in time. Particles begin at a radiation detector, and tallies are made at radiation source locations.

Biasing: a distortion of natural analog processes to achieve variance reduction in certain desired output quantities.

Body: one of the geometric entities used to construct a zone. In CG, these are geometric primitives such as spheres, boxes, etc. In CAD, a body is not a geometric primitive, and a body and a zone refer to the same entity.

B-rep: the boundary representation geometry description typically employed by CAD packages.

CAD: computer aided design, also used to refer to the boundary representation (B-rep) geometries created by CAD.

CEPXS: the Coupled Electron-Photon X-Section generation code used to create multigroup cross sections for ITS.

CG: abbreviation for Combinatorial Geometry. The method of combining simple geometric entities, such as tori and spheres, into more complicated geometry descriptions. This method is also know as Constructive Solid Geometry (or CSG). CG frequently refers to the use of bodies (such as boxes and truncated cones), while CSG frequently refers to the use of surfaces (such as planes and parabolas).

Code Options: the set of valid preprocessor definitions used to select a version of the ITS codes to be compiled.

Code Zone: unions in the CG description of an input zone are maintained internally as separate code zones for particle tracking purposes.

Collision Forcing: a biasing technique used for photons. Photons entering a zone are forced to interact with a specified probability, which may be larger or smaller than the natural interaction probability. The weight of the photon is modified accordingly.

Combinatorial Geometry: see CG.

Constructive Solid Geometry: see CG.

CSG: see CG.

CUI: the Combined User Inputs employed when running ITS with shell scripts.

Cutoff Energy: the energy below which particles are not simulated in detail. The cutoff energies may be different for electrons and photons.

CYLTRAN: the axisymmetric cylindrical material geometry ITS codes, with fully three dimensional description of particle trajectories.

CYLTRANM: the MCODES version of CYLTRAN.

CYLTRANP: the PCODES version of CYLTRAN.

Dose: energy absorbed by a material per unit mass, often used interchangably with the term "energy deposition", which is not normalized per unit mass.

Electron Trapping: see Trapping.

ETRAN: the one dimensional, single material electron/photon transport code developed by the National Institute of Standards and Technology from which the ITS codes were developed.

Forward: the solution of the forward transport problem. This simulation method is similar to the physical processes going forward in time. Particles begin at a radiation source, and tallies are made at radiation detector locations.

Group: a span of the particle energy domain within the multigroup approximation over which particles are assumed to interact with the same probabilities.

HYBRID Geometry: the geometry may be described by a combination of CAD and CG zones.

Input Body: see Body.

Input Zone: see Zone.

ITS: the Integrated TIGER Series codes, sometimes referring to only the continuous-energy versions of the codes and not the MITS codes.

ITS-CAD: the ITS and MITS codes capable of tracking particles in three dimensional CAD geometries.

KERMA: Kinetic Energy Released to MAterial is the energy deposition calculated in the absence of electron transport with the assumption of electronic equilibrium. Various assumptions can be made regarding the radiative energy that would be produced by electrons. CEPXS assumes that radiative energy is not lost from the system and is locally deposited.

Keywords: the set of words that may be used in the input deck to activate and deactivate code features.

MCODES: the ITS codes which enable transport in macroscopic electric and magnetic fields of arbitrary spatial dependence.

MITS: the Multigroup Integrated TIGER Series codes.

Multigroup: an approximation involving the discretization of the particle energy domain into groups. The approximation is only accurate if the cross sections do not vary significantly over each group.

Next Event Estimator: a biasing technique used for photons. The probability that a photon will escape without interacting is used to record an escape tally, with the weight of the tally modified accordingly, regardless of whether the photon actually escapes in the particular simulation.

Overlay: a tally structure superimposed upon a zone for calculating the subzone distribution of energy and charge deposition.

PCODES: the more elaborate ionization/relaxation model adapted from the SANDYL code.

Preprocessor: the compiler preprocessor used to apply the selection of the code version and include the necessary common block statements into the source code. The cpp program is used for preprocessing ITS.

Prmfile: the file containing CAD parameter input.

RNG: random number generator.

Russian Rouletting: a biasing technique in which particles may be eliminated with some probability. If the particle is not eliminated from the simulation, the weight of the particle is modified accordingly.

Satfile: the file containing the CAD geometry in ACIS text format.

Scaling: a biasing technique based on scaling material cross sections (larger or smaller) and correspondingly modifying the weighting of any particle exposed to the scaled cross sections.

Sub-keyword: a secondary keyword that will not be recognized without the presence of the associated primary keyword.

Subzone: a geometry entity, usually a small fraction of a zone, in which energy and charge deposition are tallied.

TIGER: the one dimensional geometry capability of the ITS codes.

TIGERP: the PCODES version of TIGER.

Trapping: a biasing technique used for electrons. Because electrons have a finite range within a material, it may not be possible for an electron to escape the zone or subzone that it is in. If it cannot escape, it is "trapped." Trapping may neglect effects of secondary photons that can escape from the zone.

XGEN: the coupled electron-photon X-section GENeration code used to create continuous-energy cross sections for ITS.

Zone: a geometry entity that consists of a single material and density.

References

[1] M. J. Berger and S. M. Seltzer, ETRAN Monte Carlo code system for electron and photon transport through extended media, CCC-107, Radiation Shielding Information Center, Computer Code Collection, Oak Ridge National Laboratory, 1968.

[2] M. J. Berger, Monte Carlo calculation of the penetration and diffusion of fast charged particles, in *Methods in Computational Physics*, edited by B. Adler, S. Fernbach, and M. Rotenberg, volume 1, pages 135–215, Academic, New York, 1963.

[3] M. B. Emmett, The MORSE Monte Carlo radiation transport code system, Technical Report ORNL-4972, Oak Ridge National Laboratory, 1975.

[4] D. P. Sloan, A new multigroup Monte Carlo scattering algorithm suitable for neutral and charged-particle Boltzmann and Fokker-Planck calculations, Technical Report SAND83-7094, Sandia National Laboratories, 1983.

[5] J. A. Halbleib and W. H. Vandevender, EZTRAN–a user-oriented version of the ETRAN-15 electron-photon Monte Carlo technique, Technical Report SC-RR-71-0598, Sandia National Laboratories, 1971.

[6] J. A. Halbleib and W. H. Vandevender, EZTRAN 2: A user-oriented version of the ETRAN-18B electron-photon Monte Carlo technique, Technical Report SLA-73-0834, Sandia National Laboratories, 1973.

[7] J. A. Halbleib and W. H. Vandevender, Nuclear Science and Engineering **57**, 94 (1975).

[8] J. A. Halbleib and W. H. Vandevender, Nuclear Science and Engineering **61**, 288 (1976).

[9] J. A. Halbleib, Nuclear Science and Engineering **75**, 200 (1980).

[10] W. Guber, J. Nagel, R. Goldstein, P. S. Mettelman, and M. H. Kalos, A geometric description technique suitable for computer analysis of both the nuclear and conventional vulnerability of armored military vehicles, Technical Report MAGI-6701, Mathematical Applications Group, Inc., 1967.

[11] E. A. Straker, J. W. H. Scott, and N. R. Byrn, The MORSE code with combinatorial geometry, Technical Report SAI-72-511-LJ (DNA 2860T), Science Applications, Inc., 1972.

[12] J. A. Halbleib and J. E. Morel, Nuclear Science and Engineering **70**, 219 (1979).

[13] J. A. Halbleib and J. E. Morel, CYLTRANP, Sandia National Laboratories, unpublished, 1981.

[14] H. M. Colbert, SANDYL: A computer code for calculating combined photon-electron transport in complex systems, Technical Report SLL-74-0012, Sandia National Laboratories, 1974.

[15] J. A. Halbleib, Sr. and W. H. Vandevender, Journal of Applied Physics **48**, 2312 (1977).

[16] L. F. Shampine, H. A. Watts, and S. Davenport, SIAM Rev. **18**, 376 (1976).

[17] K. L. Hiebert and L. F. Shampine, Implicitly defined output points for solutions of ODEs, Technical Report SAND80-0180, Sandia National Laboratories, 1980.

[18] J. A. Halbleib, ACCEPTM, Sandia National Laboratories, unpublished, 1981.

[19] J. A. Halbleib, Nuclear Science and Engineering **66**, 269 (1978).

[20] J. A. Halbleib, SPHEM, 1979, Sandia National Laboratories, unpublished.

[21] J. A. Halbleib and T. A. Mehlhorn, Nuclear Science and Engineering **92**, 338 (1986).

[22] T. A. Mehlhorn and J. A. Halbleib, Monte Carlo benchmark calculations of energy deposition by Electron/Photon showers up to 1 GeV, in *Proceedings of a Topical Meeting on Advances in Reactor Computations*, page 608, 1983, ISBN 0-89448-111-8.

[23] J. A. Halbleib, R. P. Kensek, G. D. Valdez, S. M. Seltzer, and M. J. Berger, IEEE Transactions on Nuclear Science **39**, 1025 (1992).

[24] L. J. Lorence, Jr., R. P. Kensek, and J. A. Halbleib, Transactions of the American Nuclear Society **73**, 339 (1995).

[25] L. J. Lorence, R. P. Kensek, J. A. Halbleib, and J. E. Morel, IEEE Transactions on Nuclear Science **42**, 1895 (1995).

[26] J. E. Morel, L. J. Lorence, R. P. Kensek, J. A. Halbleib, and D. P. Sloan, Nuclear Science and Engineering **124**, 369 (1996).

[27] B. C. Franke et al., Adjoint charge deposition and CAD transport in ITS, in *Proceedings of the ANS International Meeting on Mathematical Methods for Nuclear Applications*, Salt Lake City, Utah, 2001.

[28] B. C. Franke et al., ITS version 5.0: The integrated TIGER series codes, Technical report, Sandia National Laboratories, 2002, unpublished draft.

[29] J. A. Halbleib and T. A. Mehlhorn, ITS: The Integrated TIGER Series of coupled Electron/Photon Monte Carlo transport codes, Technical Report SAND84-0573, Sandia National Laboratories, 1984.

[30] MacKichan Software, Inc., MacKichan software, http://www.mackichan.com, 2003.

[31] D. Price et al., Version management with CVS, http://www.cvshome.org, 2002.

[32] Amtec Engineering, Inc., Amtec Engineering home page, http://www.amtec.com, 2003.

[33] J. A. Halbleib, J. Appl. Phys. **45**, 4103 (1974).

[34] J. A. Halbleib and W. H. Vandevender, IEEE Trans. Nucl. Sci. **NS-22**, 2356 (1975).

[35] J. A. Halbleib, Bull. Am. Phys. Soc. **26**, 1062 (1981).

[36] J. S. Hendricks, Nuclear Science and Engineering **109**, 86 (1991).

[37] G. Marsaglia, A. Zaman, and W. W. Tsang, Statist. Prob. Lett. **9**, 35 (1990).

[38] F. James, Comput. Phys. Commun. **60**, 329 (1990).

[39] P. L'Ecuyer, INFORMS J. on Computing **9**, 57 (1997).

[40] M. Matsumoto and T. Nishimura, ACM Transactions on Modeling and Computer Simulations (TOMACS) **8**, 3 (1998).

Index

DISTRIBUTION:

1 MS 0139
P. Wilson, 09902

1 MS 0139
R. K. Thomas, 09904

1 MS 0139
S. Lott, 09905

1 MS 0153
L. Hernandez, 10503

1 MS 0316
J. Castro, 09233

1 MS 0316
S. Hutchinson, 09233

1 MS 0370
M. Eldred, 09211

1 MS 0370
T. G. Trucano, 9211

1 MS 0372
J. Lash, 09126

1 MS 0376
T. Blacker, 09226

1 MS 0376
J. Fowler, 09226

1 MS 0376
J. F. Shepherd, 09226

1 MS 0378
P. F. Chavez, 09232

1 MS 0380
K. Alvin, 09142

1 MS 0417
D. Fordham, 09743

1 MS 0425
L. C. Trost, 09745

1 MS 0427
B. Bedeaux, 02134

1 MS 0427
B. Paulsen, 02134

1 MS 0481
R. Harrison, 02132

1 MS 0525
C. Bogdan, 01734

1 MS 0525
P. Plunkett, 01734

1 MS 0525
T. Russo, 01734

1 MS 0670
P. L. Dreike, 5526

1 MS 0759
H. Schriner, 04145

1 MS 0807
M. Rajan, 09328

1 MS 0828
M. Pilch, 9133

1 MS 1106
K. Mikkelson, 15342

1 MS 1106
J. Rivera, 15342

1 MS 1109
R. Benner, 09224

1 MS 1110
S. Goudy, 09224

1 MS 1137
G. D. Valdez, 06224

1 MS 1142
J. W. Bryson, 06881

1 MS 1145
T. R. Schmidt, 06880

1 MS 1146
P. J. Cooper, 06871

1 MS 1146
K. R. Depriest, 06871

1 MS 1146
P. J. Griffin, 06871

1 MS 1146
G. Harms, 06871

1 MS 1146
K. O. Reil, 06871

1 MS 1152
M. L. Kiefer, 01642

1 MS 1152
P. Mix, 06142

1 MS 1152
K. Reed, 01643

1 MS 1152
D. Seidel, 01642

1 MS 1152
C. David Turner, 01642

1 MS 1153
P. McDaniel, 15331

1 MS 1159
W. H. Barrett, 15344

1 MS 1159
C. Coverdale, 15344

1 MS 1159
V. Harper-Slaboszewicz, 15344

1 MS 1159
M. A. Hedemann, 15344

1 MS 1159
S. Jones, 15344

1 MS 1166
C. R. Drumm, 15345

1 MS 1166
W. C. Fan, 15345

1 MS 1166
G. J. Scrivner, 15345

1 MS 1166
T. Wrobel, 15345

1 MS 1167
T. Calocci, 15343

1 MS 1167
F. Hartman, 15343

1 MS 1167
L. Posey, 15343

1 MS 1167
D. Wrobel, 15343

1 MS 1170
R. Skocypec, 15310

1 MS 1176
R. Cranwell, 15313

1 MS 1179
D. E. Beutler, 15341

1 MS 1179
L. Cordova, 15341

1 MS 1179
M. Crawford, 15341

10 MS 1179
B. C. Franke, 15341

1 MS 1179
R. Hohlfelder, 15341

10 MS 1179
R. P. Kensek, 15341

10 MS 1179
T. W. Laub, 15341

1 MS 1179
J. R. Lee, 15340

5 MS 1179
L. J. Lorence, 15341

1 MS 1179
J. K. McDonald, 15341

1 MS 1179
S. D. Pautz, 15341

1 MS 1179
J. L. Powell, 15341

1 MS 1182
R. Kaye, 15335

1 MS 1182
T. Lockner, 15335

1 MS 1186
T. Brunner, 01674

1 MS 1186
P. J. Christenson, 01674

1 MS 1186
T. Mehlhorn, 01674

1 MS 1188
J. Wagner, 15311

1 MS 1193
J. Maenchen, 01645

1 MS 1196
D. L. Fehl, 01677

1 MS 1196
T. Sanford, 01677

1 MS 1207
S. A. Dupree, 05935

1 MS 1219
T. P. Wright, 05923

1 MS 1423
P. Miller, 01118

1 MS 9036
W. P. Ballard, 08200

1 MS 9042
A. Ting, 08752

1 MS 9221
A. M. F. Lau, 08517

1 MS 9403
A. Antolak, 08773

1 MS 9405
K. L. Wilson, 08770

1 MS 9018
Central Technical Files, 8945-1

2 MS 0899
Technical Library, 9616