

SAND91-1634 • UC-405
Supersedes SAND84-0573 dated November 1984
Unlimited Release
Printed March 1992

ITS Version 3.0: The Integrated TIGER Series of Coupled Electron/Photon Monte Carlo Transport Codes

J. A. Halbleib, R. P. Kensek, T. A. Mehlhorn, G. D. Valdez, S. M. Seltzer,
M. J. Berger

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

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

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

Printed in the United States of America. This report has been reproduced directly from the best available copy.

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

Prices available from (615) 576-8401, FTS 626-8401

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

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND91-1634
(Revision of SAND84-0573)
Unlimited Release
Printed March 1992

ITS Version 3.0:
The Integrated TIGER Series of
Coupled Electron/Photon
Monte Carlo Transport Codes

J. A. Halbleib and R. P. Kensek
Simulation Technology Research Division
and
T. A. Mehlhorn, Target Physics Analysis Division
Sandia National Laboratories
Albuquerque, NM 87185

G. D. Valdez
Applied Methods, Albuquerque, NM 87108

S. M. Seltzer and M. J. Berger
Ionizing Radiation Division, Center for Radiation Research
National Institute of Standards and Technology, Gaithersburg, MD 20899

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 machine-portable utility that emulates the basic features of the CDC UPDATE processor, the user selects one of eight codes for running on a machine of one of at least four major vendors. The ease with which this utility 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 maximized by employing the best available cross sections and sampling distributions, and the most complete physical model for describing the production and transport of the electron/photon cascade from 1.0 GeV

down to 1.0 keV. Flexibility of construction permits the more sophisticated user to tailor the codes to specific applications and to extend the capabilities of the codes to more complex applications through simple update procedures. Version 3.0, the latest version of ITS, contains (1) major improvements to the physical model, (2) additional variance reduction via both internal restructuring and new user options, and (3) expanded input/output capabilities. Moreover, the general user friendliness of the software has been substantially enhanced through increased internal error checking and improved code clarity, maintainability, and portability.

ACKNOWLEDGEMENTS

We wish to express our gratitude to Pedro Andreo and his colleagues at the Karolinska Institute and the University of Stockholm, to Charles M. Snell and Hsiao-Hua Hsu of the Los Alamos National Laboratory, to Nino R. Pereira and Steve Gorbics of Berkeley Research Associates, and to Ronald L. Weitz and Maximo S. Lazo of Science Applications International Corporation for their generous cooperation in acting as "friendly users" for this revision of the Integrated TIGER Series; to J. E. Powell, W. Beezhold, and J. R. Lee whose encouragement and cooperation were essential to the success of this work; to Bernadette Kirk of the Radiation Shielding Information Center, ORNL, for her extensive testing of the revised system on an IBM main frame; and to the Defense Nuclear Agency for their partial funding of the development of the improved cross-section data at the National Institute of Standards and Technology. Lastly, we wish to thank Berta Armijo for assisting in the preparation of this document.

This work was supported by the U. S. Department of Energy under Contract No. DE-AC04-76DP00789.

Preface to Version 3.0

Since its initial release, 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 seen major improvements in physical accuracy, new capabilities, variance reduction, and user friendliness.

This document is the first revision of SAND84-0573, which was the sole documentation for the original version of ITS. Separate memoranda dated April 29, 1987 and December 11, 1987 accompanied the releases of Versions 2.0 and 2.1, respectively. These memoranda described new capabilities only and in no way nullified any part of the original document. However, the magnitude of the changes in Version 3.0 have necessitated the current revision. These changes are best described in terms of the two major software components of the ITS system, the source code for the integrated cross-section generators and the source code for the integrated Monte Carlo codes. This preface is primarily directed toward those users who have used Versions 1.0 through 2.1, in order to briefly outline all important operational changes and new capabilities relative to the original version of this document. This discussion is of less importance to new users who are starting with Version 3.0, since a more detailed discussion is included within the main body of this revision.

For completeness, we also include a discussion of the relatively minor modifications covered in the two above-mentioned memoranda. We will not discuss in any detail the extensive new trapping of illegal or inconsistent input data, the numerous other improvements that were made to the internal structure of the codes, or the improvements to the UPEML processor. The latter are covered in the latest documentation of the processor as referenced herein.

Monte Carlo Codes

The Monte Carlo codes were extensively rewritten. The core modification, which led to important gains in variance reduction for a large class of applications, was a reorganization of certain portions of the logic for electron production and transport. A premium has been placed on the earliest possible terminal processing of certain trapped electrons and electrons that are rejected through Russian Roulette. A simultaneous benefit was improved protection against the possibility of obtaining misleading results in certain applications through overbiasing.

The output of line radiation leakage is much improved. The line radiation is no longer included in the continuum photon spectra. Instead, a complete and separate accounting of line radiation is included. The leakage for each line is given explicitly, in integral form as well as differential in angle. Furthermore, each line is identified by the initial and final states of the atomic transition as well as the transition energy.

These line radiation edits have been extended to photon flux as well. Earlier versions only gave the energy spectra of the volume-averaged fluxes. As part of the upgrade, the particle fluxes were expanded to include angular distributions and doubly differential (energy and angle) distributions so that the flux outputs are now as

comprehensive as the particle leakages. An ancillary result of this work was development of a utility routine to facilitate obtaining doubly differential distributions of other user-coded outputs.

With the addition of an option for photon-only transport, the ITS system now includes a very fast and sophisticated set of photon transport codes. This option is easily selected via the standard keyword execution input and is much faster than coupled transport (up to an order of magnitude for test problems). It should be used for problems with source photons for applications in which (1) the finite ranges of the secondary electrons are unimportant and (2) radiation by those electrons is unimportant. Photon-only transport should also be useful for parameter studies where approximate results are acceptable and the above conditions are not rigorously satisfied, or for low-energy photon sources where the ITS codes, because they are continuous-energy codes, have the advantage of highly accurate simulation of line radiation.

Bona fide automatic subzoning has been implemented in the CYLTRAN codes. The subzoning in Version 1.0 was not authentic because costly full particle tracking was applied to subzones, even though their boundaries were not material discontinuities. In Version 3.0, these boundaries are used only for scoring spatially-dependent quantities such as charge and energy deposition (and sometimes flux) within the corresponding input zone. The power of this option lies in the fact that it not only substantially reduces the burden of input data, as it did in Version 1.0, but it also can result in significant reductions in run time and memory. Moreover, with the inclusion of azimuthal dependence, the automatic subzoning has been extended to three dimensions. Perhaps even more important, because of the more costly tracking logic, this fully three-dimensional automatic subzoning logic has been extended to input zones of the ACCEPT codes that consist of simple bodies such as, cylinders, spheres, and rectangular parallelepipeds of arbitrary orientation.

The next-event estimator for photon leakage in the TIGER and CYLTRAN codes has been extended to the ACCEPT codes. This option is now 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 keyword, NEXT-EVENT-ESCAPE.

The logic permitting biased bremsstrahlung production has been modified so that this biasing can be based on any one of the materials in the cross-section set. Previously, the biasing had to be based on the first material, a technicality that often required the cross sections to be rerun with the materials reordered.

The energy range for the cross-section sets in Version 1.0 was too restrictive. In Version 2.0, both the cross-section generator and the Monte Carlo codes were modified in order to permit generation and use of cross sections that span any arbitrary energy range between 1.0 GeV and 1.0 keV. This improvement is useful either (a) because the problem to be solved simply requires electron transport over an unusually large energy range or (b) because a series of calculations in which there is a large variation in the source energies can be carried out more efficiently using a single cross-section set. Numerous modifications were required to permit the use of the improved cross sections that are discussed below.

Finally, some routines have been completely rewritten. For example, the three-dimensional combinatorial-geometry routines of the ACCEPT codes were rewritten for both improved efficiency and improved clarity. Similarly, the enhanced ionization/relaxation subroutine employed in the P codes was rewritten for improved clarity alone, with some small penalty in execution time.

Cross-Section Generator Codes

Compared to Version 1.0, Version 3.0 contains major improvements to the physical model. These are discussed in detail in Appendix M and include the following:

- 1) There is a completely new set of bremsstrahlung cross sections differential in photon energy that replace cross sections largely based on the Born Approximation.
- 2) There is a much improved algorithm for sampling electron collisional energy loss from the Landau/Blunck-Leisegang straggling distribution.
- 3) New collisional stopping powers have been implemented. They make use of an improved data base of mean excitation energies and an improved algorithm for calculating the density-effect correction for conductors and non-conductors. The radiation-loss stopping powers have also been updated using the improved bremsstrahlung cross sections.
- 4) The latest data base of photon cross sections from the National Institute of Standards and Technology has been implemented. These cross sections allow us to include coherent scattering of photons in the Monte Carlo for the first time. Moreover, these cross sections include binding effects for both coherent and incoherent scattering.

The archaic Fortran from the pre-integration generators that had been retained in Version 1.0 was completely rewritten using the BLOCK-IF structure of the Fortran 77 standard. The parameterization of arrays is now much more extensive, and literal constants have been parameterized to facilitate the use of double precision. Changes to the names of common variables across modules were eliminated, so that only a single literal copy of each common block appears in the source code for the program library. Finally, error checking of input data has been expanded.

TABLE OF CONTENTS

	Page
1. Introduction	11
1.1 History of the TIGER Series	11
1.2 Overview of the ITS Code Package	13
1.3 Overview of the Document	15
2. Operation	16
2.1 Syntax	16
2.1.1 Syntax for Machine and Code Selection Input	16
2.1.2 Syntax for Code Execution Input	17
2.2 Running a Cross-Section Generating Code	18
2.2.1 Select a Cross-Section Generating Code	18
2.2.2 Execute the Cross-Section Generating Code	19
2.3 Running a Monte Carlo Code	22
2.3.1 Select a Monte Carlo Code	22
2.3.2 Execute the Monte Carlo Code	24
2.4 Suggestions for Efficient Operation	42
2.5 Output	44
2.6 Availability	45
3. Machine Portable Update Processor - UPEML	46
3.1 Structure of the Program Libraries	46
3.1.1 The SOURCE Library	46
3.1.2 The Binary Old Program Library (OLDPL)	48
3.2 The Creation Run	48
3.3 Correction Runs	48
4. Monte Carlo Program File - ITS	51
4.1 Eight Codes	51

4.2 Four Machines	51
4.3 Free-Format Input	53
4.4 Parameterization	54
4.5 Embellishments	55
4.6 Additional Restructuring Subsequent to Version 1.0	56
5. Cross-Section Generating File - XGEN	57
6. Cross-Section Data File - XDATA	59
7. Concluding Remarks	60
Appendix A: Installation	61
A.1 Overview	61
A.2 Step-by-Step Installation Guide	61
Appendix B: Sample Input Streams	64
B.1 Sample Input for Running TIGER Code	66
B.2 Sample Input for Running CYLTRAN Code	67
B.3 Sample Input for Running ACCEPT Code	68
B.4 Sample Input for Running TIGERP Code	70
B.5 Sample Input for Running CYLTRANP Code	72
B.6 Sample Input for Running ACCEPTP Code	74
B.7 Sample Input for Running CYLTRANM Code	77
B.8 Sample Input for Running ACCEPTM Code	79
Appendix C: TIGER Codes - Geometry	81
C.1 Problem Geometry	81
C.2 Conventions for Escaping Particles	81
Appendix D: CYLTRAN Codes - Geometry	82
D.1 Problem Geometry	82
D.2 Conventions for Escaping Particles	84

Appendix E: ACCEPT Codes - Geometry	85
E.1 Problem Geometry	85
E.1.1 Body Definition	85
E.1.2 Specification of Input Zones	89
E.1.3 Volume Specification	92
E.1.4 Material Specification	92
E.2 Input Data	92
E.2.1 Body Data	92
E.2.2 Input Zone Data	93
E.2.3 Volume Data	95
E.2.4 Material Data	95
Appendix F: P Codes	96
Appendix G: M Codes	97
Appendix H: Biasing Options and Variance Reduction	99
H.1 Trapped Electrons	99
H.2 Scaling of Bremsstrahlung Productions	100
H.3 Scaling the Probability for Electron Impact Ionization	100
H.4 Forced Photon Collisions (TIGER and CYLTRAN Codes)	100
H.5 Photon Path Length Stretching (ACCEPT Codes)	101
H.6 Zone Dependent Electron Cutoff Energies	101
H.7 Next-Event Estimator for Photon Escape	102
Appendix I: Plotting Capability	103
I.1 CYLTRAN and ACCEPT Code Geometries	103
I.2 Electron and Positron Trajectories (M Codes)	104
I.3 Implementation of Plot Packages	104
Appendix J: Statistics	105

Appendix K: Notes on Job Control Language (JCL)	107
K.1 Generic Creation Runs	107
K.2 Generic Correction Runs	108
K.3 Examples of IBM JCL	111
Appendix L: Automatic Subzoning	115
Appendix M: Cross-Section Improvements in Version 3.0	118
M.1 Bremsstrahlung Cross Sections	118
M.2 Electron and Positron Stopping Powers	119
M.3 Energy-Loss Straggling	120
M.4 Photon Cross Sections	121
M.5 Incoherent Photon Scattering	121
M.6 Photon Coherent Scattering	122
Appendix N: Random Number Generators	123
N.1 Range	123
N.2 Access to the Seed	123
N.3 Reproducibility	123
N.4 Cycle Length	124
N.5 Portable Random Number Generator	124
References	125

1. Introduction

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. It is based primarily on the ETRAN model¹, which combines microscopic photon transport with a macroscopic random walk² for electron transport. Emphasis is on simplicity of application without sacrificing the rigor or sophistication of the physical model.

1.1 History of the TIGER Series

Table I chronicles the development of the TIGER series, beginning with the EZTRAN³ and EZTRAN2⁴ 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.

TIGER⁵, CYLTRAN⁶, and ACCEPT⁷ 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^{8,9}.

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 TIGERP¹⁰ and CYLTRANP¹¹ codes, we added the more elaborate ionization/relaxation model from the SANDYL code¹² 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 CYLTRANM¹³, we combined the collisional transport of CYLTRAN with transport in macroscopic electric and magnetic fields of arbitrary spatial dependence using a Runga-Kutta-Fehlberg algorithm¹⁴ to integrate the Lorentz force equations. An important modification of this algorithm¹⁵ made possible the development of the ACCEPTM code¹⁶, which combines the collisional transport of the ACCEPT code with macroscopic field transport. SPHERE¹⁷ and SPHEM¹⁸ were two special purpose codes that were restricted to multiple concentric spherical shells without and with macroscopic field transport, respectively.

Table I: 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
ITS 1.0	Dec 84	Yes	all ^b
ITS 2.0	Mar 87	Yes	all ^b
ITS 2.1	Dec 87	Yes	all ^b

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

^b The integrated system is not restricted to a single dimensionality. The member codes are listed in Table II.

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.

1.2 Overview of the ITS Code Package

ITS consists of four essential elements:

- 1) XDATA --- The electron/photon cross-section data file
- 2) XGEN --- The cross-section generation program file
- 3) ITS --- The Monte Carlo program file
- 4) UPEML --- A machine portable update emulator

The heart of ITS is the Monte Carlo program file. Although the structure of this file is discussed in detail below, we point out its salient features here for those who only wish to run the codes. The combined program library file was obtained by integrating the eight codes of Table II 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 has been added to the ACCEPT code. Those individual codes appearing in Table I, but not in Table II, are of a more specialized nature than the others and are no longer supported since their function is duplicated by at least one of the ITS codes. Each of the eight member codes will run on any of four machines -- CRAY, CDC, VAX (double precision) or IBM (double precision). Although the codes have only been tested on these four machines, the use of Fortran 77 (i.e., American National Standard FORTRAN, ANSI X3.9-1978) should facilitate installation on other machines as well. Additional cross-section data and associated logic allows transport from 1.0 keV to 1.0 GeV¹⁹ 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 has resulted in a very simple and user-friendly input scheme. Integration of the various codes has resulted in the availability of additional common options for each code. Also, a general restart option has been added. In an attempt to conform to modern programming practices, a complete line-by-line rewrite of the codes was carried out with emphasis on implementing the top-down

BLOCK-IF structure of Fortran 77. Finally, options are available for plotting the problem geometry in the CYLTRAN and ACCEPT codes, and, in the case of the M codes, for plotting electron and positron trajectories in regions where macroscopic fields exist.

Table II: ITS Member Codes

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

From top to bottom, the codes grouped by row in Table II 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 P codes (Appendix F), and the M codes (Appendix G), respectively.

Since the initial release,²⁰ 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.

This revision for Version 3.0 is the first revision of the original documentation, SAND84-0573,²⁰ of Version 1.0 of the ITS system.²⁰⁻²³ Separate memoranda dated April 29, 1987 and December 11, 1987 accompanied the releases of Versions 2.0 and 2.1, respectively. These memoranda described new capabilities only, relative to Version 1.0, and in no way nullified any part of the original document. However, the magnitude of the changes in Version 3.0 have necessitated the current revision. For completeness, this revision also covers the relatively minor modifications covered in the two above-mentioned memoranda, as well as improvements to the UPEML processor. To recapitulate, this revision represents the sole documentation of Version 3.0 of the ITS system with an abbreviated discussion of Version 3.0 of the UPEML processor.

Finally, we wish to emphasize that while the source code of the program libraries explicitly provides for running the codes on the main frames of four vendors, Version 3.0 of the ITS system has been extensively tested only on Cray/UNICOS and VAX/VMS at Sandia National Laboratories, and on IBM/MS-DOS at the Radiation Shielding Information Center, Oak Ridge National Laboratory. Version 3.0 has not been tested at all on CDC machines. Nevertheless, because users have successfully ported Versions up through 2.1 to many machines, including workstations and PCs, that run under a variety of operating systems, we believe Version 3.0 will prove to be very portable as well.

1.3 Overview of the Document

This document is arranged in such a way that for relatively routine problems the casual user need only acquaint himself with Section 2 once the ITS package has been installed on the host computer as described in Appendix A. Section 2 contains all of the information necessary for the routine application of the member codes. Section 3 discusses the machine portable update emulator. This section should be read by those users who are not familiar with the syntax and operation of the CDC UPDATE processor,²⁴ or its equivalent, since the emulator employs that syntax to select individual codes from the combined program library and, if desired, to modify them. Subsequent sections may be ignored altogether, or their perusal may be delayed until such time as a more intimate knowledge of the structure of ITS becomes necessary. Sections 4, 5, and 6 discuss the structure of the Monte Carlo program file, the cross-section generating program file, and the cross-section data file, respectively. Concluding remarks are given in Section 7.

The ITS system has been extensively tested throughout its development. Nevertheless, for an effort of this magnitude -- involving multiple machines, multiple codes, and numerous input options -- it is impossible to be completely confident that no bugs exist (indeed, this is never strictly possible with any piece of software). Consequently, we solicit feedback from users on any difficulties they may encounter and on any suggestions they may have for improving future versions of the system.

2. Operation

It is assumed that ITS has been properly installed. This is accomplished once and for all as described in Appendix A. Two basic steps are required for solving any given problem:

- (1) Generate cross sections by running the cross-section code
- (2) Run the Monte Carlo code

A successful run of either a cross-section code or a Monte Carlo code requires a job stream that sequentially performs the following two tasks:

- (1) Selects a machine and the code to be run on that machine
- (2) Executes the code

Each task requires a separate input file in the job input stream.* Using the sample input streams for the eight Monte Carlo member codes shown in Appendix B as examples, we will discuss these inputs in the order in which the four tasks are performed. Before doing so, however, we must describe the syntax of these inputs.

2.1 Syntax

There are two kinds of syntax that must be followed. First, there is the syntax for the input to the update emulator that is used to select a machine and the code to be run on that machine. Secondly, there is the syntax for the the free-format input used by the selected code at execution time. These will be discussed separately. In order to do so, we adopt the convention that character strings enclosed in square brackets ([]) represent information that is to be supplied by the user.

2.1.1 Syntax for Machine and Code Selection Input --- This input is used by the machine portable processor, UPEML,^{25**} to process either the cross-section generating program file, XGEN, or the Monte Carlo program file, ITS. The input always begins with a command line that

* Although the Job Control Language for accomplishing these tasks will very much depend on the local operating system, we provide some guidance for setting up the job control stream in Appendix K. This appendix also includes a discussion of all input/output units.

** The CDC UPDATE processor²⁴ may also be used, as may any update processor that recognizes the syntax of this input and the update directives appearing in the XGEN and ITS SOURCE files (see Sec. 3.1). In the case of the CDC UPDATE processor, the function of the command line is replaced by appropriate parameters in the UPDATE command of the CDC Job Control Language.

indicates what processes should be performed, what input files to look for, and which output files to create. The simplest command line to initiate processing of the program file is:

F,P

where the "F" signifies that a compile file is to be created for use by the Fortran compiler, while the "P" signifies that a binary old program library (OLDPL) for either the cross-section generation program file or the Monte Carlo program file (as described in Appendix A) exists as the file "oldpl" for this run. The compile file will be written to the file "cmpfil". Additionally, other parameters as discussed in the UPEML manual²⁵ can be included on this line. For example, an "I" signifies that the correction run information will be read from the file "srcdk", rather than from the default input device (typically unit 5). These lines should begin in column 1. We assume here that the input stream of update directives required by UPEML for selecting code and machine is in the job stream. The first directive must always be

*IDENT,[name]

where [name] is any alphanumeric string, beginning with a letter, of up to 7 characters chosen by the user to identify this correction set. No imbedded blanks are permitted, and the asterisk must be in column 1. This *IDENT directive is followed by one or more *DEFINE directives. The syntax for the *DEFINE directives is

*DEFINE,[keyword]

where [keyword] is replaced by any of the keywords that select the machine or the code to be run on that machine. Examples will be given later in this section and in Appendix B.

Using this set of directives as input, UPEML performs what is technically referred to as a correction run, as opposed to the creation runs discussed in Appendix A. Each of the files XGEN and ITS consists of multiple versions (for different machines) of multiple codes that have been integrated into a single file. In these correction runs UPEML provides us with a simple mechanism for selecting a particular version of a particular code.

We have just described the normal input required for selecting the machine and the code to be run on that machine. However, once the user becomes familiar with the Fortran coding of the XGEN and ITS files, he can use UPEML to modify that coding in any way he wishes. The procedures for doing this are described in Sec. 3.3.

2.1.2 Syntax for Code Execution Input --- Having selected a code and machine, the user must supply the necessary input for executing that code for his particular problem. The input scheme employed in the original unintegrated versions of the member codes of ITS was completely discarded in Version 1.0 in favor of a free-format scheme that offers many advantages to the user. Column counting is no longer necessary; numerical data are merely separated by one or more blank spaces.

Options are selected through a system of descriptive keywords, making the input file much more self-descriptive. The primary keywords are order independent. Input is minimized by making maximum use of defaults -- that is, options and/or parameter values assumed by the code when one or more keywords do not appear in the input stream (e.g., if the input file for XGEN is empty, 1.0-MeV cross sections for aluminum are calculated). Furthermore, the keyword scheme permits the selection of options via standard input that, in the original unintegrated versions, could only be activated via the CDC UPDATE processor or its equivalent (e.g., UPEML). More internal processing makes input rules less rigid. Finally, extensive internal error checking traps many input errors immediately, thereby reducing the need for time-consuming debugs and tracebacks.

The general syntax for code-execution input is:

```
[keyword] [parameter(1)] [parameter(2)] ...
```

That is, the user specifies a particular keyword and may follow it by numerical parameters that are associated with that keyword. Though the primary keywords are order independent, secondary keywords must follow the primary keyword to which they are subordinate before the specification of a new primary keyword. However, the secondary keywords for a given primary keyword are also order independent. If fewer than the maximum allowable number of parameters for a given keyword are entered, those not entered will be set equal to zero. Consequently, in this free-format input, only trailing parameters may be omitted. Numerous examples of this syntax will be given below.

Finally, an asterisk appearing in column one of a line of input will cause that line to be regarded as a comment, unless that line represents mandatory input required by a particular keyword. This feature permits the user to insert comments for improving the readability of his input stream or to temporarily comment out certain input rather than deleting it altogether.

2.2 Running a Cross-Section Generating Code

First, select a cross-section generating code; then compile, link/load, and execute.

2.2.1 Select a Cross-Section Generating Code --- This selection entails running UPEML on the saved program library (OLDPL) for XGEN (Appendix A) in order to create the input file for the Fortran 77 compiler. The input stream required to accomplish this when UPEML is run would look like:

```
{eor}  
F,P  
*IDENT,[name]  
*DEFINE,[keyword(1)]  
*DEFINE,[keyword(2)]  
{eor}
```

The string {eor} specifies an end-of-record mark throughout this document. UPEML will look for the OLDPL on the file "oldpl"; the OLDPL should be assigned to this name in the job control stream. Legitimate keywords are listed in Table III. The directive

*DEFINE,PCODES

selects the coding that will calculate the special cross sections required for the P codes; without this directive, coding for calculating the standard cross sections is obtained by default. Other examples of this UPEML input may be found between the first and second {eor}s in each of the input streams shown in Appendix B.

Table III: Keywords Used in the Input for the Update Emulator, UPEML, When Processing XGEN

<u>Machine Selection</u>	<u>Code Selection</u>
CRAY	PCODES
CDC	
VAX	
IBM	

UPEML will summarize the output statistics for this run on the logical unit designated as the system output device (typically unit 6). A compile file is generated as the file "cmpfil" for subsequent processing by the Fortran compiler.

2.2.2 Execute the Cross-Section Generating Code --- We can now compile and link/load this compile file, and the result will be the executable image of the cross-section program XGEN. This file can then be executed to create cross-section input for the Monte Carlo program. Program XGEN looks for the cross-section and atomic data file XDATA as Fortran unit 9; XDATA should be assigned to this device in the Job Control Language.

In addition, the user must provide problem specific input. Table IV lists the primary keywords used for input to the cross-section generating program along with their default values and associated secondary keywords. A typical example of their use can be seen between the second and third {eor}s of the input stream shown in Sec. B.1 of Appendix B. In most cases only the primary keywords TITLE, MATERIAL, and ENERGY will be needed.

The following is an alphabetic listing under which detailed instructions on the function and usage of each keyword are given:

1. ECHO
Syntax: ECHO [parameter(1)]
Example: ECHO 1

Default: no echo

If "ECHO 1" is inserted in the input stream, all subsequent card images will be echoed to the terminal or to the job output (whichever is appropriate).

2. 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.

Table IV: Summary of Keywords Used for Input in the Execution of the Cross-Section Generating Codes

<u>KEYWORD</u>	<u>DEFAULT</u>
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/mixtures)
DENSITY	normal density (g/cc) (pure elements only)
DENSITY-RATIO	1.0
SUBSTEP	internal
TITLE	no title
STEP	8
PRINT-ALL	abbreviated cross-section tables will be printed
ECHO	off

3. 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. To override these defaults or to construct compound materials the following secondary keywords associated with this primary keyword may be used.

- i) **element symbol(s)**
 Syntax: element symbol [parameter(1)]
 Example: TA 0.25 Al 0.75
 Default: no default; required
 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 fractions must sum to 1.0.
- ii) **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.
- iii) **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.
- iv) **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).
- Example for compound target - CH
 MATERIAL C .9225 H .0775
 DENSITY .87
- v) **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).

vi) **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.²

4. **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.

5. **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.²

6. **TITLE**

Syntax: TITLE
[parameter(1)]

Example: TITLE
Cross sections for 5 MeV electrons in Tantalum

Default: No title
Indicates that the next line in the input file will contain a title of up to 80 columns in length for the cross-section run.

The output from the execution of the cross-section generating code is a file containing all the cross sections and sampling distributions for both electrons and photons that will be needed by the Monte Carlo code. This file is associated with Fortran unit 11.

2.3 Running a Monte Carlo Code

First, select a machine and the Monte Carlo code to be run on that machine; then compile, link/load, and execute.

2.3.1 Select a Monte Carlo Code --- This selection entails running UPEML on the saved program library for ITS (Appendix A) in order to create the input file for the Fortran 77 compiler. The input stream required to accomplish this when UPEML is run would look like:

```

(eor)
F,P
*IDENT,[name]
*DEFINE,[keyword(1)]
*DEFINE,[keyword(2)]
(eor)

```

UPEML will look for the OLDPL on the file "oldpl"; the OLDPL should be assigned to this name in the job control stream. Legitimate keywords are listed in Table V. Note that in Version 3.0, the keyword DOUBLE need no longer be explicitly defined when running on VAX or IBM machines. Other examples of this UPDATE input can be found between the third and fourth {eor}s in the examples given in Appendix B.

Table V: Keywords Used in the Input for the Update Processor, UPEML, When Processing ITS

<u>Machine Selection</u>	<u>Code Selection</u>	<u>Options</u>
CRAY	TIGER	PCODES
CDC	CYLTRAN	MCODES
VAX	ACCEPT	PLOTS
IBM		RNG1

There are a few rules and restrictions on the use of these keywords. One and only one of the keywords TIGER, CYLTRAN, or ACCEPT must always be defined. At the present time, PCODES or MCODES, but never both, must be defined in order to select one of the P codes or M codes, respectively, from Table II. The use of PLOTS is restricted to the CYLTRAN and ACCEPT codes; the plot utility is discussed in more detail in Sec. 2.3.2. Invocation of the machine portable random number generator via the RNG1 keyword is discussed in Appendix N.

Additional UPEML input is required for the M codes. Subroutine BFLD must compute the cartesian components of the electric and/or magnetic fields at an arbitrary location within a zone that has been defined as having these fields (see keyword GEOMETRY in Sec. 2.3.2 for CYLTRANM and ACCEPTM). A sample version of Subroutine BFLD is included in the M codes. The user must replace or modify this routine so that it will calculate the fields for his particular problem. To do this he will probably need to use the *DELETE and *INSERT directives described in Sec. 3.3. Examples of this additional UPEML input are found between the third and fourth {eor}s of Secs. B.7 and B.8 of Appendix B.

The user may also wish to include the necessary UPEML input for modifying the Fortran 77 PARAMETER statements. These modifications are usually optional, but may be required when running complex problems on machines with limited fast memory. Moreover, the modifications may significantly reduce computing costs where the charging algorithm depends on memory usage. Indeed, PARAMETER changes will affect CPU

times on some computing systems. Definitions of the Fortran 77 PARAMETERS have been placed at the top of the ITS file for the convenience of the user, and the PARAMETER statements themselves appear immediately thereafter. On execution, the ITS member codes provide the user with a comparison of memory requirements vs allocations in order to assist him in making the modifications. Examples of Fortran 77 PARAMETER modifications are shown in Sec. B.6 of Appendix B.

UPEML will summarize the output statistics for this run on the logical unit designated as the system output device (typically unit 6). A compile file is generated as the file "cmpfil" for subsequent processing by the Fortran compiler.

2.3.2 Execute the Monte Carlo Code --- We can now compile and link/load the compile file and the result will be the executable image of the Monte Carlo program that has been selected. This file can then be executed to solve the electron/photon transport problem. Program ITS looks for a cross-section file that has been previously created by the cross-section generating code. This cross-section file is associated with Fortran unit number 11; this assignment should be made in the job control stream.

In addition, the user must provide problem specific input. Table VI lists the primary keywords used for input to the Monte Carlo program along with their default values. In order to facilitate their use, they have been organized into the five functional categories shown in the table. Typical examples of their use can be seen after the fourth {eor}s of the input streams shown in Appendix B.

The following is an alphabetic listing under which detailed instructions on the function and usage of each of the keywords are given:

1. BATCHES

Syntax: BATCHES [parameter(1)]

Example: BATCHES 20

Default: 10 batches

Number of batches of primary particles to be run. The total number of histories is divided into [parameter(1)] batches containing an equal number of source particles in order to obtain estimates of statistical uncertainties. Accuracy of the estimates degrades substantially for fewer than 10 batches. Although increasing the number of batches improves this accuracy (for a given total number of histories), it also increases the overhead (run time).

Table VI: Summary of Primary Keywords Used for Input in the Execution of the Monte Carlo Codes

KEYWORD	DEFAULT
	**** GEOMETRY ****
GEOMETRY	required
	**** SOURCE ****
ELECTRONS or PHOTONS	electron source
ENERGY or SPECTRUM	1.0 MeV monoenergetic
POSITION	point source at origin for TIGER and ACCEPT codes or on axis at the minimum-z boundary of the problem geometry for the CYLTRAN codes
DIRECTION	monodirectional source in positive-z direction
CUTOFFS	electrons: 5% of maximum energy photons: 0.01 MeV
	**** OUTPUT OPTIONS ****
ELECTRON-ESCAPE	off
ELECTRON-FLUX	off
PHOTON-ESCAPE	off
PHOTON-FLUX	off
PULSE-HEIGHT	off
	**** OTHER COMMONLY USED OPTIONS ****
ECHO	off
TITLE	blank title
HISTORIES	1000 histories
PLOTS ^a	no plots
TRAP-ELECTRONS	no check on trapping for electrons with energies greater than the zone-dependent cutoff
SCALE-BREMS	natural bremsstrahlung production
	**** OTHER RARELY USED OPTIONS ****
BATCHES	10 batches
DUMP	off
RESTART	no restart
PRINT-ALL	print only final batch output
RANDOM-NUMBER	machine default
NEW-DATA-SET	1 run
SCALE-IMPACT	20% of bremsstrahlung scaling if SCALE-BREMS is used; otherwise, natural electron impact ionization
NEXT-EVENT-ESCAPE	off unless PHOTON-ESCAPE keyword is used
NO-COHERENT	photon coherent scattering is simulated
NO-INCOH-BINDING	binding effects in incoherent scattering are included

^a CYLTRAN and ACCEPT codes

2. CUTOFFS

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

Example: CUTOFFS 0.10 0.01

Default: [parameter(1)] -- global electron cutoff energy equals 5% of maximum source energy

[parameter(2)] -- photon cutoff energy equals 0.01

[parameter(1)] and [parameter(2)] are the global cutoff energies (MeV) at which electron and photon histories are terminated, respectively.

This electron cutoff energy is the global cutoff; the cutoffs in individual zones may be raised above this value as described under the GEOMETRY keyword. When an electron falls below the cutoff energy in a given zone, a check is first made to see if it is trapped in the sense described under keyword 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. For either of 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.

With the default dimensioning in the codes, the electron cutoff must be greater than or equal to the maximum energy for which electron cross sections have been calculated divided by 244 (assuming default STEP parameter in Sec. 2.2.2) or 0.001 MeV, whichever is the larger. For photons the residual energy is assumed to be deposited on the spot. The photon cutoff must be ≥ 0.001 .

The default energy range spanned by the electron cross-section sets in Version 1.0 was sometimes too restrictive, and an expanded energy range could only be obtained through relatively cumbersome code modifications using the UPEML processor. Starting with Version 2.0, both the cross-section generator and Monte Carlo codes were modified in order to permit generation and use of cross sections that span an arbitrary energy range between 1.0 GeV and 1.0 keV. This improvement is useful either (a) because the problem to be solved simply requires electron transport over an unusually large energy range or (b) because a series of Monte Carlo calculations in which there is a large variation in the source energies can be carried out more efficiently using a single cross-section set. In order to invoke this capability, the UPEML processor must be used to increase Fortran parameters INX and INMAX in the cross-section generator and Monte Carlo codes, respectively. An example of these Fortran parameter modifications is shown in Sec. B.6 of Appendix B, even though the modification is not necessary for that particular problem. The default value of both parameters, which must be multiples of 8, is 64. INX can be increased until the electron cross-section energy grid extends down to or below the cutoff value desired by the user (≥ 1 keV) in the Monte Carlo runs. The minimum value of INMAX to be used in

the Monte Carlo codes and the minimum value of the cross-section energy list are given in the output of the cross-section generator as NMAX and T(NMAX+1), respectively.

3. DIRECTION

Syntax: DIRECTION [parameter(1)] (TIGER codes)
 or
 DIRECTION [parameter(1)] [parameter(2)] (CYLTRAN and ACCEPT codes)

Example: DIRECTION 45.0 (TIGER codes)
 DIRECTION 45.0 135.0 (CYLTRAN and ACCEPT codes)

Default: reference direction is positive z direction

This keyword defines the source reference direction. [parameter(1)] and [parameter(2)] (not needed for TIGER codes) are the spherical polar angles θ and ϕ , respectively, in degrees, that define the reference direction. The meaning of source parameters specified by the POSITION and DIRECTION keywords is illustrated in Fig. 1. In the absence of either of the following secondary key-

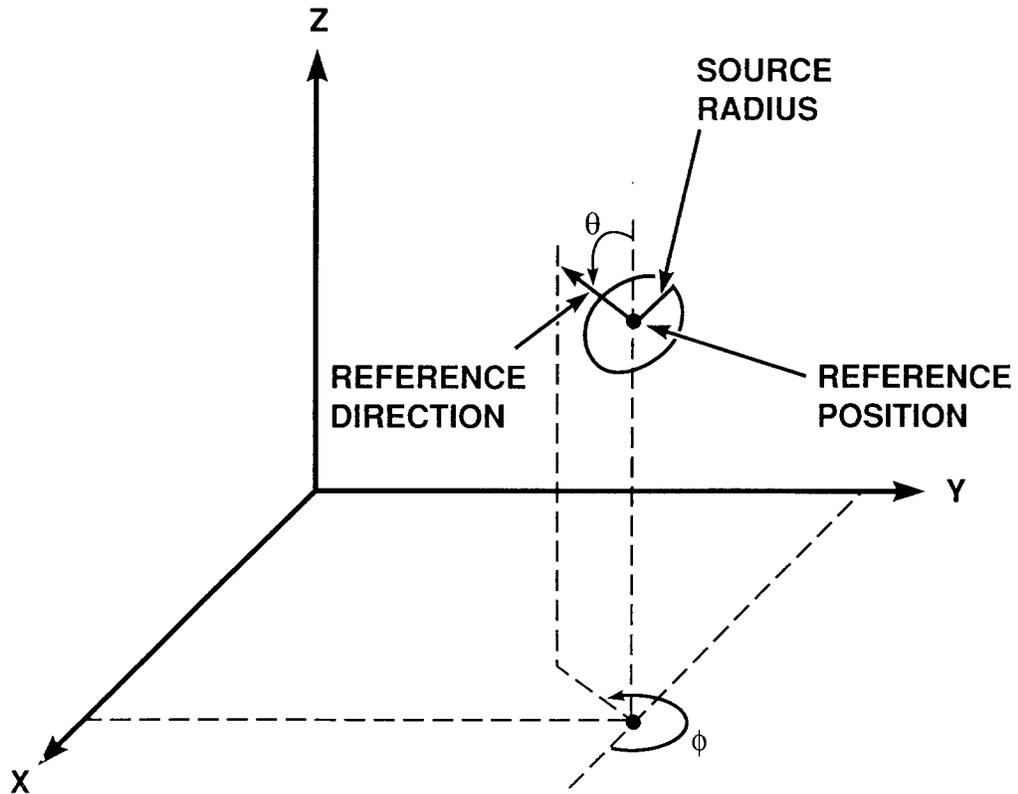


Figure 1: Illustration of Source Parameters Defined by the POSITION and DIRECTION Keywords

words associated with this primary keyword, the source angular distribution defaults to a monodirectional source defined by the reference direction.

ISOTROPIC

Syntax: ISOTROPIC [parameter(1)]

Example: ISOTROPIC 45.0

Default: no isotropic distribution

Defines angular distribution of source particles as being isotropic with respect to the reference direction. If the parameters associated with the DIRECTION keyword are left blank, the default reference direction is assumed. The distribution is truncated at an angle of [parameter(1)] in degrees. The default value for [parameter(1)] is 90.0; i. e., a 2π isotropic source.

COSINE-LAW

Syntax: COSINE-LAW [parameter(1)]

Example: COSINE-LAW 45.0

Default: no cosine-law distribution

Defines angular distribution of source particles as being proportional to the cosine of the angle with respect to the reference direction. If the parameters associated with the DIRECTION keyword are left blank, the default reference direction is assumed. The distribution is truncated at an angle of [parameter(1)] in degrees. The default value for [parameter(1)] is 90.0; i. e., a 2π cosine-law source.

4. DUMP

Syntax: DUMP

Example: DUMP

Default: no dump

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

5. ECHO

Syntax: ECHO [parameter(1)]

Example: ECHO 1

Default: no echo

If "ECHO 1" is inserted in the input stream, all subsequent card images will be echoed to the terminal or to the job output (whichever is appropriate).

6. ELECTRONS

Syntax: ELECTRONS

Example: ELECTRONS

Default: electron source if neither ELECTRONS nor PHOTONS keyword is used

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

7. ELECTRON-ESCAPE

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.

i) **NBINE**

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

Example: NBINE 5 USER

0.8 0.5 0.2 0.1 0.05

Default: 10 bins of equal width

[parameter(1)] is the number of energy bins; if omitted, 10 will be assumed. If [keyword] is not specified, equal width bins are used. Choices for [keyword] are:

a) LOG -- Logarithmic grid spacing with

$$E_{i+1} = 2^{-1/8} E_i.$$

b) USER -- User defined energy grid: code will then read the lower bound (MeV) for the number of energy bins specified by [parameter(1)] in descending order as in the above example. The maximum lower bound must be less than the maximum source energy.

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

ii) **NBINT**

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

Example: NBINT 5 USER

10.0 30.0 90.0 135.0 180.

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

[parameter(1)] is the number of polar angle bins; if omitted, 18 will be assumed. If [keyword] is not specified, equal width bins from 0.0 to 180.0 degrees will be defined. In this case, [parameter(1)] must be even for the TIGER and CYLTRAN codes in order to unambiguously distinguish emission from the transmission and reflection surfaces as defined in Appendices C and D. The only choice for [keyword] is:

a) USER -- User defined angle grid: code will then read the upper bounds (degrees) for the number of angle bins specified by [parameter(1)] in ascending order as in the above example. The largest value must be 180.0, and the value 90.0 must also be included for the TIGER and CYLTRAN codes in order to unambiguously distinguish emission from the transmission and reflection surfaces.

iii) **NBINP** (CYLTRAN and ACCEPT only)

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

Example: NBINP 6 USER

10.0 30.0 75.0 120.0 200.0 360.0

Default: 1 bin of 360.0 degrees

[parameter(1)] is the number of azimuthal angle bins; if omitted, 1 will be assumed. If [keyword] is not specified, equal width bins from 0.0 to 360.0 degrees will be defined. The only choice for [keyword] is:

- a) USER -- User defined angle grid: code will then read the upper bounds (degrees) for the number of angle bins specified by [parameter(1)] in ascending order as in the above example. The largest value must be 360.0.

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 using the UPEML processor to increase the value of the parameter IKMAZ (IKPMAZ for photon escape) as defined near the top of the ITS file.

Note that the primary keyword alone, with no other parameters or keywords, will result in the calculation of electron escape using the default bin structures.

8. ELECTRON-FLUX

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

Example: ELECTRON-FLUX 3 5

Default: electron flux is not 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 Appendix L. If either parameter is omitted or 0, flux will be calculated in all zones. 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 GEOMETRY) for zones [parameter(1)] through [parameter(2)] are all equal. The following are secondary keywords associated with this primary keyword that describe the bin structure used in tallying electron flux.

i) NBINE

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

Example: NBINE 5 USER

0.8 0.5 0.2 0.1 0.05

Default: 10 bins of equal width

[parameter(1)] is the number of energy bins; if omitted, 10 will be assumed. If [keyword] is not specified, equal width bins are used. Choices for [keyword] are:

- a) LOG -- Logarithmic grid spacing with $E_{i+1}=2^{-1/8}E_i$.
- b) USER -- User defined energy grid: code will then read the lower bounds (MeV) for the number of energy bins specified by [parameter(1)] in descending order as in the above example.

The maximum lower bound must be less than the maximum source energy.

For either of these tertiary keywords, the user must insure that the lowest energy of the energy grid is less than or equal to the common zone-dependent cutoff energy (if less than, the grid will be truncated) for the zones where flux is to be calculated.

ii) **NBINT**

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

Example: NBINT 5 USER

10.0 30.0 70.0 100.0 180.0

Default: 6 bins of equal width from 0.0 to 180.0 degrees

[parameter(1)] is the number of polar angle bins; if omitted, 6 will be assumed. If [keyword] is not specified, equal width bins from 0.0 to 180.0 degrees will be defined. The only choice for [keyword] is:

a) **USER** -- User defined angle grid: code will then read the upper bounds (degrees) for the number of angle bins specified by [parameter(1)] in ascending order as in the above example. The largest value must be 180.0.

iii) **NBINP** (CYLTRAN and ACCEPT only)

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

Example: NBINP 6 USER

10.0 30.0 75.0 120.0 200.0 360.0

Default: 1 bin of 360 degrees

[parameter(1)] is the number of azimuthal angle bins; if omitted, 1 will be assumed. If [keyword] is not specified, equal width bins from 0.0 to 360 degrees will be defined. The only choice for [keyword] is:

a) **USER** -- User defined angle grid: code will then read the upper bounds (degrees) for the number of angle bins specified by [parameter(1)] in ascending order as in the above example. The largest value must be 360.0.

WARNING: Because azimuthal scoring of flux radiation is only rarely requested, the azimuthal dimension of the flux 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 using the UPEML processor to increase the value of the parameter IKFMAZ (IKFMZP for photon flux) as defined near the top of the ITS file.

Note that the primary keyword alone, with no other parameters or keywords, will result in the calculation of flux in every input zone of the problem (excluding the escape zone of the ACCEPT codes) using the default bin structures.

9. **ENERGY**

Syntax: ENERGY [parameter(1)]

Example: ENERGY 2.0

Immediately after the keyword line there must follow a series of [parameter(1)] lines, one for each input zone, containing either [parameters(2)] through [parameter(11)] for CYLTRAN and CYLTRANP, or [parameter(2)] through [parameter(12)] for CYLTRANM. For CYLTRAN and CYLTRANP, 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, the number of z subzones, the input-zone-dependent electron cutoff energy in MeV (see Sec. H.6 of Appendix H for more detail) and the photon forced interaction probability (see Sec. H.4 of Appendix H for more detail), respectively. In the case of CYLTRANM, the macroscopic field flag is inserted ahead of the cutoff energy. 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. These parameters are discussed in more detail in Appendix D. When the trailing fields for the field flag, electron cutoff and photon forced-interaction probability are left blank, there will be no fields in the given zone, the electron cutoff will be set to the global electron cutoff (normal -- see primary keyword CUTOFFS), and there will be no forcing of photon interactions (normal).

10c. ACCEPT codes (see Appendix E)

```
Syntax: GEOMETRY [parameter(1)] [parameter(2)]
        Combinatorial geometry description of problem bodies and
        input zones as described in Appendix E
        [parameter(3)] [parameter(4)] [parameter(5)]
                                                (ACCEPT or ACCEPTP)
        or
        [parameter(3)] [parameter(4)] [parameter(5)]
        [parameter(6)]
                                                (ACCEPTM)
```

```
Example: GEOMETRY 0 1 (ACCEPTM, 4 input zones)
        Combinatorial geometry description of problem bodies and
        input zones as described in Appendix E
        3 0 0.03 0.4
        1 1
        2
        0
```

Default: no default; required for ACCEPT codes

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 cm³:

- (a) 0 (normal and default) causes the volumes to be set internally to 1.0.
- (b) 1 causes the code to read the volumes from the input stream as described in Appendix E.
- (c) 2 requires that the user provide the necessary logic for computing the volumes at the appropriate place in Subroutine JOGEN.

Tracking debug is turned off or on according to whether [parameter(2)] is set equal to 0 (default) or not, respectively. What follows this keyword line is first the combinatorial geometry description of the problem bodies and input zones as described in Appendix E. Immediately after this information there must follow a series of lines, one for each input zone, containing either [parameters(3)] through [parameter(5)] for ACCEPT and ACCEPTP, or [parameter(3)] through [parameter(6)] for ACCEPTM. For ACCEPT and ACCEPTP, these parameters specify the material, the input-zone-dependent electron cutoff energy in MeV (see Sec. H.6 of Appendix H for more detail) and the photon stretching factor (see Sec. H.5 of Appendix H for more detail), respectively. In the case of ACCEPTM, the macroscopic field flag is inserted ahead of the cutoff energy. This flag specifies the macroscopic fields that are present in the given input 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). When the trailing fields for the field flag, electron cutoff and photon stretching factor are left blank, there will be no fields in the given zone, the electron cutoff will be set to the global electron cutoff (normal, see primary keyword CUTOFFS), and there will be no stretching of photon interactions (normal).

11. HISTORIES

Syntax: HISTORIES [parameter(1)]

Example: HISTORIES 10000

Default: 1000 histories

Number of primary particle histories to be followed.

12. 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 single exception to the rule that the primary keywords are order independent. The cross-section file generated according to Sec. 2.2 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. Obviously, running multiple problems will require more run time than running a single problem and may lead to slower turnaround due to an assignment of lower priority by the operating system.

Note: We have also discovered another completely unrelated, but valuable, use for this keyword for the more sophisticated user. If the user has modified the software in such a way that additional data must be read after all other keywords for the run have been read, this keyword can simply be inserted ahead of this

data to prevent its processing as standard keyword input. However, to insure graceful termination, the modification

*DELETE,INPUT.920

should be included in the input to the UPEML correction run so that the Monte Carlo code will not expect to read execution input for a subsequent run.

13. NEXT-EVENT-ESCAPE

Syntax: NEXT-EVENT-ESCAPE

Example: NEXT-EVENT-ESCAPE

Default: Next-event estimator for photon escape is turned off unless it has been automatically activated via the PHOTON-ESCAPE keyword.

This keyword activates the next-event estimator as a method of variance reduction for integral photon leakage when the PHOTON-ESCAPE keyword is not used. It is automatically activated for integral and differential photon leakage when the PHOTON-ESCAPE keyword is used. The next-event method for photon leakage is discussed further in Sec. H.7 of Appendix H.

14. 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.

15. 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.

16. PHOTONS

Syntax: PHOTONS

Example: PHOTONS

Default: electron source

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

PHOTRAN

Syntax: PHOTRAN

Example: PHOTRAN

Default: The simulation will make use of fully coupled electron/photon transport.

This secondary keyword eliminates all electron transport, simulating photon-only transport. For many applications involving low-energy photon sources, the ranges of secondary electrons and the production of radiation by those electrons are negligible, so that photon transport alone is sufficient.

This ITS approximation has the advantage over some alternative photon transport methods in that it otherwise rigorously simulates the production and transport of line radiation. This option can greatly reduce run time relative to that for the fully coupled default software.

17. PHOTON-ESCAPE

same as ELECTRON-ESCAPE, except that line radiation output is given separately from the continuum, and uncollided radiation (photon sources) is not included in the default option

18. PHOTON-FLUX

same as ELECTRON-FLUX, except that line radiation output is given separately from the continuum, and uncollided radiation (photon sources) is not included in the default option

19. PLOTS

This keyword is only valid for the CYLTRAN and ACCEPT codes, assuming that an appropriate graphics package is available on the machine on which these codes are to be run. The plotting capability of ITS is discussed in detail in Appendix I. The choice between the following two usages depends on which of the member codes of ITS has been selected.

19a. CYLTRAN

Syntax: PLOTS [parameter(1)] [parameter(2)] ... [parameter(4)]

Example: PLOTS 0.0 5.2 -3.2 2.0

Default: no plots

This keyword is used to plot the problem geometry for any of the CYLTRAN codes and, for CYLTRANM, the electron and positron trajectories in zones where macroscopic fields are defined. Use of this keyword will produce a ρ -z plot of that portion of the problem cylinder bounded by [parameter(1)] through [parameter(4)] which define the minimum ρ , the maximum ρ , the minimum z and the maximum z, respectively, in cm. If the parameters are left blank, the entire problem cylinder will be plotted. The following secondary keyword defines which electron and positron trajectories are to be plotted in the case of CYLTRANM.

ORBITS

Syntax: ORBITS [parameter(1)]

Example: ORBITS 10

Default: Electron and positron trajectories associated with the 5th, 10th 15th, ... source particle of the first batch will be plotted.

Electron and positron trajectories associated with source particles of the first batch that are multiples of [parameter(1)] are to be plotted in those zones for which macroscopic fields have been defined. A blank for [parameter(1)] will cause all electron and positron trajectories of the first batch to be plotted.

19b. ACCEPT codes

Syntax: PLOTS [parameter(1)]
 [parameter(2)] [parameter(3)] ... [parameter(7)]

.

.

.

Example: PLOTS 3
 -10.0 10.0 -10.0 10.0 90.0 180.0
 -10.0 10.0 -10.0 10.0 180.0 90.0
 -10.0 10.0 -10.0 10.0 0.0 0.0

Default: no plots

Use of this keyword will produce [parameter(1)] parallel projections of the body specification as given under the GEOMETRY keyword. [parameter(2)] through [parameter(7)] are repeated [parameter(1)] times on separate lines. [parameter(6)] and [parameter(7)] specify the spherical polar angles ϕ and θ , respectively, in degrees that define the direction from which the geometry is to be viewed. [parameter(2)] through [parameter(5)] specify the minimum x, the maximum x, the minimum y, and the maximum y, respectively, in cm of the plotted projection. If the PLOTS keyword is used, all parameters are required; there are no defaults. The following secondary keyword defines which electron and positron trajectories are to be plotted in zones where macroscopic fields are defined and is therefore valid only for the ACCEPTM code. The trajectories will only be plotted on the final projection.

ORBITS

Syntax: ORBITS [parameter(1)]

Example: ORBITS 10

Default: Electron and positron trajectories for the 5th, 10th, 15th, etc., source particle of the first batch will be plotted.

Electron and positron trajectories associated with source particles of the first batch that are multiples of [parameter(1)] are to be plotted in those zones for which macroscopic fields have been defined. A blank for [parameter(1)] will cause all electron and positron trajectories of the first batch to be plotted.

20. POSITION

This keyword defines the reference position of the source. The choice between the following two usages depends on which of the member codes of ITS has been selected.

20a. TIGER codes

Syntax: POSITION [parameter(1)]

Example: POSITION 2.0

Default: source at z=0.0

[parameter(1)] is the z coordinate in cm.

20b. CYLTRAN and ACCEPT codes

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

Example: POSITION 0.0 0.0 2.0

Default: point source at x=0.0, y=0.0 and z=minimum-z plane of
problem cylinder (CYLTRAN codes)

or

point source at origin (ACCEPT codes)

[parameter(1)] through [parameter(3)] are the x, y and z coordinates, respectively, in cm. When this keyword is used without parameters, the default reference position is used, but now a finite radius source may be specified by the secondary keyword, RADIUS, that is associated with this primary keyword.

RADIUS

Syntax: RADIUS [parameter(1)]

Example: RADIUS 3.5

Default: source radius equals 0.0

This keyword defines the radius in cm of a disk source with the reference position at its center. The normal to the disk will be the reference direction as defined by the DIRECTION keyword. The meaning of source parameters specified by the POSITION and DIRECTION keywords is illustrated in Fig. 1.

WARNING: The keywords DIRECTION, POSITION, and RADIUS must be defined such that any sampled source position will fall within some defined input zone (excluding the escape zone for the ACCEPT codes).

21. 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 keyword causes the cumulative results from all batches to be written to the output file.

22. PULSE-HEIGHT

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 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.

i) **NBINE**

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

Example for monoenergetic source energy of 2 MeV:

```
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 total 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). When [keyword] is USER, [parameter (1)] is the number of bin energies to be read. The only choice for [keyword] is:

- a) **USER** -- User defined energy grid: 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 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.

23. **RANDOM-NUMBER**

Syntax: RANDOM-NUMBER

[parameter(1)]

Example: RANDOM-NUMBER

```
4265641542
```

Default: 0

[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. It 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. The format for [parameter(1)] will depend on the random number routines that are being used. See Sec. 4.2 for further discussion of issues concerning random number routines.

24. RESTART

Syntax: RESTART [parameter(1)]

Example: RESTART 4

Default: no restart

This keyword is used to signal that a problem is to be restarted from the output of the batch indicated by [parameter(1)]. A dump must have been written and saved (see keyword DUMP) on a previous run. The dump file must be assigned to Fortran unit 14 for the restart run. Furthermore, the batch size (number of histories divided by the number of batches) on the restart run must be the same size as those for which the dump file was written in order to permit the accurate computation of statistical uncertainties.

25. SCALE-BREMS

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

Example: SCALE-BREMS 100.0 3

Default: natural probability of bremsstrahlung production

This keyword is used to scale bremsstrahlung production. [parameter(1)] is used to scale bremsstrahlung production so as to increase the photon population without increasing the number of primary histories. A more detailed discussion of this option can be found in Appendix H. [parameter(2)] is the index of the material, according to the order in which the materials are read, on which the scaling is to be based -- material number 3 in the example. The default is material number 1. The following secondary keyword associated with this primary keyword is used to control the number of secondary electrons generated by this increased population of photons.

i) ELECTRON-RR

Syntax: ELECTRON-RR [parameter(1)]

Example: ELECTRON-RR 0.1

Default: (normal) 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 omitted or 0.0, the natural number of electrons will be followed (the number produced if SCALE-BREMS had not been used).

26. SCALE-IMPACT

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 factor for electron impact ionization will be set to 20% of [parameter(1)] of that keyword (normal).

This keyword is used to scale electron impact ionization. [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. A more detailed discussion of this option can be found in Appendix H.

27. SPECTRUM

Syntax: SPECTRUM [parameter(1)]

Example: SPECTRUM 6

1.0 0.8 0.45 0.2 0.05 0.0

0.9 0.7 0.5 0.3 0.1 0.0

Default: monoenergetic source

This keyword signals that the energies of the source particles are to be sampled from a spectrum. As shown in the above example, the next line(s) of input must contain [parameter(1)] values of the normalized cumulative distribution of source particles in descending order from 1.0 to 0.0, and the next line(s) must contain the corresponding energies (MeV) in descending order. In the above example the source energies will be sampled from the spectrum shown in Fig. 2. Note that the spectrum may now extend below the appropriate cutoff energy as defined by the keyword CUTOFFS. If the sampled energy of a source particle is below the cutoff, the particle is rejected, and an accounting of the number and energy of rejected source particles is given in the output.

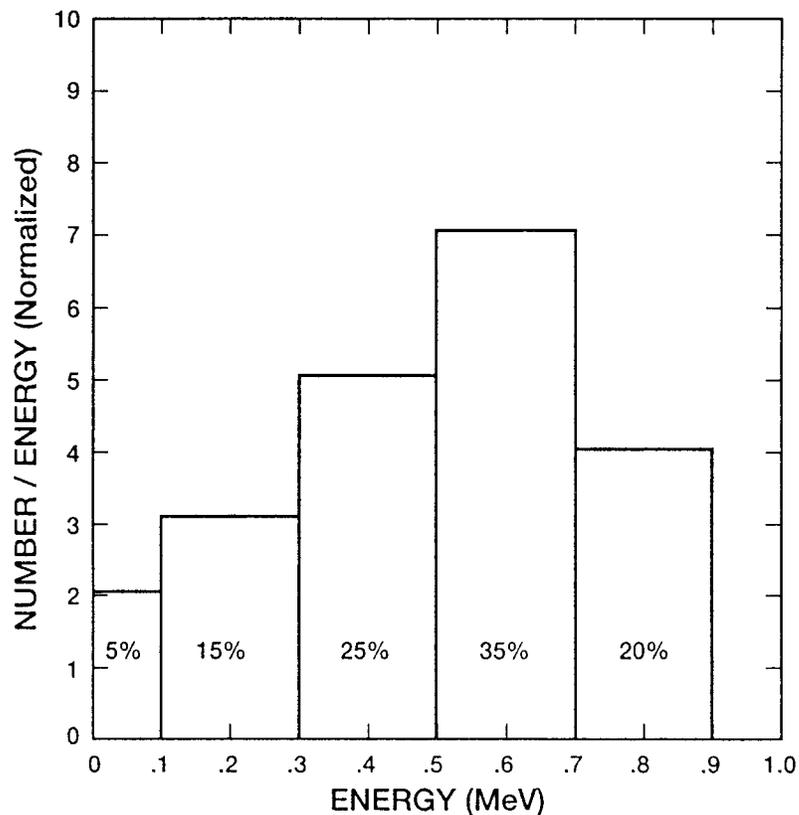


Figure 2: Differential Source Spectrum from Which the Cumulative Distribution in the Example Given under the SPECTRUM Keyword Was Obtained

28. TITLE

Syntax: TITLE
[parameter(1)]

Example: TITLE
1.0 MEV BREMSSTRAHLUNG PRODUCTION IN TA

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 also be used as the title on any plots that are generated.

29. TRAP-ELECTRONS

Syntax: TRAP-ELECTRONS [parameter(1)]

Example: TRAP-ELECTRONS 0.2

Default: The electron trapping energy for each zone will be set equal to the zone dependent cutoff energy.

In certain problems where only electrons that cross certain boundaries are important, this keyword may be employed to significantly reduce running time. [parameter(1)] is the global trapping energy in MeV. The zone-dependent trapping energy for each zone will be set equal to the greater of this value or the zone-dependent cutoff energy. However, if electron flux output has been requested for a given zone (see keyword ELECTRON-FLUX), the trapping energy for that zone will, if necessary, be reduced to the cutoff energy for that zone.

If an electron in a given zone has an energy less than the trapping energy for that zone, even though the electron energy may be greater than the cutoff energy for that zone (see keyword GEOMETRY), and if that electron cannot escape from the zone or, in some cases, subzone (see below), its history will be terminated via on-the-spot deposition of its charge and energy. No further secondary production (e.g., bremsstrahlung production) will occur. This electron-trapping option is based on subzone boundaries in the TIGER codes, axial and radial subzone boundaries in the CYLTRAN codes, and zone boundaries in the ACCEPT codes. More detail on this option can be found in Sec. H.1 of Appendix H.

2.4 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 using the update processor, UPEML, as described in Sec. 3.3 to modify the PARAMETER statements. The definitions of these integer parameters and the corresponding PARAMETER statements are located at the very beginning

of the Monte Carlo program file (see also Sec. 4.4). We do not wish to exaggerate the necessity of this procedure since, in most cases, we have made conservatively large choices for the default values.

Immediately after a particular Monte Carlo member code has successfully processed the keyword input, it prints out a virtually complete 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. Moreover, the modifications may significantly reduce computing costs where the charging algorithm depends on memory usage. Indeed, PARAMETER changes will affect CPU times as well on some computing systems.

CAUTION: 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:

- a. 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 (see Sec. 2.5). 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.
- b. 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.
- c. 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.

- d. 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 biasing options can markedly increase the efficiency of certain calculations. Specific examples of these are discussed under keywords SCALE-BREMS and SCALE-IMPACT. The stretching or contraction of photon mean free paths, or forcing of photon interactions is discussed under keyword GEOMETRY. Some of these are discussed in even more detail in Appendix H. Users are warned, however, that the reckless and indiscriminate use of biasing procedures can lead to misleading results.

2.5 Output

In addition to certain diagnostic information, the default output consists of:

- a. Energy and number escape fractions (leakage) for electrons, unscattered primary photons, and scattered photons
- b. Charge and energy deposition profiles
- c. An explicit statement of energy conservation
- d. For the TIGER codes, the number and energy electron boundary currents for each layer

These data are sufficient to confirm the general partitioning and conservation of charge and energy. In very unusual situations where very high cutoff energies are specified, a decrement of charge and energy may be unaccounted for because particles escaping with energies below the cutoffs are not tallied.

In addition to the default output, a number of optional outputs may be selected through the use of the appropriate keywords. These are:

- a. Escape fractions that are differential in energy for electrons, scattered continuum photons, and line radiation
- b. Escape fractions that are differential in angle for electrons, scattered continuum photons, and line radiation
- c. Coupled energy and angular distributions of escaping electrons, scattered continuum photons, and line radiation
- d. Volume averaged fluxes that are differential in energy for electrons, scattered continuum photons, and line radiation for selected regions of the problem geometry
- e. Volume averaged fluxes that are differential in angle for electrons, scattered continuum photons, and line radiation for selected regions of the problem geometry
- f. Coupled energy and angular distributions of volume averaged fluxes of electrons, scattered continuum photons, and line radiation for selected regions of the problem geometry
- g. A pseudo-pulse-height distribution for a selected region of the problem geometry -- for example, the region corresponding to active detector elements.

By scattered continuum we mean that these photon flux and escape outputs do not include unscattered primary photons. The latter can often be calculated analytically, and, if not excluded from these outputs, will often dominate the distributions, masking the contribution from the scattered photons for which the Monte Carlo simulation is required.

For the standard codes, line radiation output can be obtained for either one ($Z < 13$) or three ($Z \geq 13$) K-fluorescent lines of the highest-atomic-number element in each material. For the P codes, all lines for each element in the problem can be obtained. These lines can only be obtained provided that their energies are greater than 1.0 keV and greater than the photon cutoff energy. In every case the lines are identified by the atomic shell transition and the fluorescent energy.

The more sophisticated user will be able to modify existing outputs and add additional ones through the correction run procedures described in Sec. 3.3.

Except for an initial diagnostic table containing accounting information on the various particle types, 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 in percent. Details of the method used to obtain these statistical data are given in Appendix J.

2.6 Availability

The European community should request the ITS software package from:

OECD NEA Data Bank
91191 Gif-sur-Yvette
CEDEX
FRANCE
Téléphone 69 08 49 12
Télécopieur 69 41 39 65
Earn/Bitnet: DBMAIL@FRNEAB51

All others can obtain it from:

Radiation Shielding Information Center
Oak Ridge National Laboratory
P. O. Box 2008
Oak Ridge, Tennessee 37831-6362
USA
Commercial Phone No. 615-574-6176
FTS Phone No. 624-6176
FAX 615-574-9619
BitNet: PDC@ORNLSTC
Internet: PDC@EPIC.EPM.ORNL.GOV

3. Machine Portable Update Processor - UPEML

The ITS system contains several cross-section and Monte Carlo transport codes combined within the two program libraries, XGEN and ITS. The desired code is extracted from one of these combined libraries for execution. A CDC UPDATE processor²⁴ or its equivalent, Version 3.0 of UPEML,²⁵ is used to perform this extraction as well as to modify the code. UPEML is the machine-portable UPDATE emulator included in this package, and Version 3.0 has additional capabilities and error checking relative to the initial version. The structure of the program libraries as well as the process of extraction and modification of the Fortran coding are explained below.

Before proceeding, however, a word about the names of the input/output I/O files used by the UPEML processor is in order. Table I of Ref. 25 gives the default file names associated with various command line parameters (e.g., file "srcdk" for parameter I), and we have used these default file names throughout this document. Alternatively, the user may override a default file name on the command line itself by equating the parameter to some other file name. For example, instead of just using I on the command line, he or she could use I=mychoice. File "mychoice" would then replace the default file, "srcdk".

3.1 Structure of the Program Libraries

The ITS and XGEN program libraries exist in two forms:

- 1) card images - the SOURCE library, and
- 2) an UPDATE-processed binary file - the OLDPL (Old Program Library).

3.1.1 The SOURCE Library --- The SOURCE library (e.g., the XGEN file or the ITS file as obtained from the magnetic tape of the ITS system package) differs from a normal Fortran program that can be passed directly to the compiler in that it contains additional directives, recognized by the UPDATE processor, that make possible the integration of multiple machine versions of multiple codes. These directives consists of: *DECK,[deckname] and *COMDECK,[comdeckname] deck name identifiers; *CALL,[comdeckname] instructions to copy common decks into a specified location; and *IF DEF,[keyword], *IF -DEF,[keyword], and *ENDIF conditional identifiers. The directives are interspersed with standard Fortran card images to create the desired library structure. A brief example of a SOURCE library follows. The numbers to the left correspond to the explanations below.

```
(2)      *COMDECK,COM1
          COMMON /BLOCK1/ X(100), Y(100)
(1)      *DECK,MAIN
          PROGRAM SAMPLE1
```

```

(2)      *CALL, COM1
          DIMENSION Z(100)
          X(1)=0.0
          DO 100 I = 1, 100
(3)      *IF DEF, OPT1
          Y(I) = X(I)+FUNC1(I)
(3)      *ENDIF
(3)      *IF DEF, OPT2
          Y(I) = X(I)+COS(X(I))
(3)      *ENDIF
          X(I) = X(I)+0.025
          100 CONTINUE
          END
(1)      *DECK, FUNC1
          FUNCTION FUNC1(J)
(2)      *CALL, COM1
          FUNC1 = 1.0+(TAN(X(J))**2
          RETURN
          END

```

- (1) Notice that both the main program and the associated function have *DECK identifiers to give them unique names. This is a feature of the ITS package; each program, subroutine, and function has a unique deck name*.
- (2) The *COMDECK names a common deck. A common deck can be inserted anywhere into the program file when requested by an associated *CALL statement. Here, the common block named BLOCK1, and identified as common deck COM1, will be inserted into the main program at the location of the *CALL, COM1 statement.
- (3) The means of combining multiple codes and multiple machine versions into a single library is demonstrated by the use of the *IF DEF, [keyword]/*ENDIF structure in this example. If OPT1 is chosen via a *DEFINE, OPT1 statement in the input deck of the correction run (see Sec. 3.3), the lines of intervening code between the *IF DEF, OPT1 and the *ENDIF are included into the compile file. Similarly, if *DEFINE, OPT2 is specified, the alternate coding is included. A statement of *IF -DEF, OPT1 would cause the intervening code to be included only if OPT1 is not defined. More complex logic structures are created by nesting *IF DEF, and *IF -DEF statements to achieve the equivalents of "and" and "and not" constructs.

* The advantage of this is that resequencing after code modification will only change the line identifiers of decks that are actually modified. This will minimize the changes necessary in the update streams used in correction runs as described in Sec. 3.3.

3.1.2 The Binary Old Program Library (OLDPL) --- The installation process described in Appendix A takes the ITS and XGEN SOURCE files and creates an equivalent binary image called the old program library (OLDPL). The initial UPEML execution of the installation process is called a "creation run" since the OLDPL that will be used in all future program executions is created. Other than the binary form of the OLDPL, the primary difference between the OLDPL and the SOURCE library is that each line in the OLDPL is uniquely labeled with a deck name and an associated deck sequence number. For example, the program card in the example is the first card after the *DECK,MAIN statement and is identified as MAIN 00002. Similarly, the next card, the DIMENSION statement, is identified as MAIN 00003. Note that the *DECK,MAIN card has itself been identified as MAIN 00001. Each line in a deck or common deck is identified in this way. The coding can then be modified as described in Section 3.3 by inserting and deleting cards at positions identified by the appropriate deck or common deck names and sequence numbers. The *IF DEF file structure information is retained in this binary format and is applied by using the appropriate set of *DEFINE,[keyword] statements in the input deck to the correction run as described in Sec. 3.3.

3.2 The Creation Run

The installation of the ITS and XGEN files involves the one-time creation of binary OLDPLs for each of these files. Once these libraries are created, they should be permanently stored on disk. We do not recommend editing the SOURCE libraries and then remaking the OLDPLs! The use of the UPEML preprocessor, in addition to allowing us to combine the eight codes in the TIGER series into a single package, permits the standardization of a particular release (Version number). That is, regardless of the host machine and user location, if ITS is installed as described, every copy of a particular version of the program library will have identical deck names, common deck names, and sequence numbers. This standardization will simplify the distribution of future modifications by the authors, both for correcting errors and for expanding program capabilities. Furthermore, if errors are detected by users, they can report the exact location of these errors to the authors, thus facilitating maintenance of the ITS system.

3.3 Correction Runs

Once the OLDPL has been created, the actual Fortran card images to pass to the compiler are obtained in what is termed a "correction run". At a minimum, this correction run requires input to define the machine and code keywords that describe the desired program. This information is contained on a series of *DEFINE,[keyword] cards, and examples of such input can be found in Appendix B. In addition to configuring the program through these definitions, the program can simultaneously be modified using appropriate combinations of the following update directives:

```
*IDENT,[name]
*INSERT,[identifier]
...
```

```
*DELETE,[identifier]
...
*BEFORE,[identifier]
...
```

The [identifier] is the appropriate deck (or common deck) name and sequence number for a card image contained in the OLDPL. The syntax for combining these two fields is such that a card that is internally identified as MAIN 00002 is identified in the input deck as "MAIN.2". The *INSERT directive is used to insert the Fortran card images that follow the directive into the program after the card image identified in the directive. The *BEFORE directive performs a similar function, except that the card images that follow the directive are inserted before the card image identified in the directive. The *DELETE directive deletes the card image identified in the directive and inserts the Fortran card images following the directive (if any) in its place. A range of program library card images can be deleted via the directive, *DELETE,MAIN.2,MAIN.5, which says to delete card images two through five inclusive from deck MAIN and replace them by the Fortran card images that follow the directive (if any). A directive consisting of */ followed by at least one space can be used to insert comments into the input stream for the UPDATE processor; these comments will be ignored by the processor. As discussed in Appendix A, a listing of all the card images included in the OLDPL along with their deck name (or common deck name) and sequence number identifiers is requested by using the "L" parameter on the command line of the creation run.

The input command line required for performing a correction run with UPEML is:

F,P

The "F" parameter denotes that a Fortran compile file is to be written to the file "cmpfil". The "P" parameter denotes that this is a correction run and that a previously created OLDPL is available as the file "oldpl". The addition of an "L" qualifier (comma-delimited) generates a summary of all operations performed (e.g., INSERTs, DELETES, DEFINE options, etc.) on the file "listfl".

The essential function of the correction run is the selection of a particular member code and a particular machine. When one of the M codes has been selected, the user will also find it necessary to include within the correction run the required logic for evaluating the magnetic and/or electric fields at a given position, as discussed in Sec. 2.3.1, by replacing or modifying the logic of Subroutine BFLD. An example of a CYLTRANM correction set can be found in Sec. B.7 of Appendix B.

The UPEML correction run provides the more sophisticated user with a much more powerful tool than is required for the simple functions that have been discussed so far. The analog nature of the Monte Carlo procedure, the completeness with which the ITS system describes the radiation transport, and the flexibility of construction of the ITS system make it possible for the user to significantly extend the capabilities of the system through the proper use of the correction run.

As the user becomes more familiar with the system, he will very likely begin using the correction run to customize individual member codes and the system itself to his own particular applications and his own particular style of operation.

4. Monte Carlo Program File - ITS

The heart of ITS is the Monte Carlo program file. Indeed, the system takes its name from the name of this file. Within this file we have integrated the Fortran coding of eight Monte Carlo codes. Each of these codes provides a detailed description of the production and transport of the electron/photon cascade from 1.0 GeV down to 1.0 keV.

4.1 Eight Codes

Using the procedure described in Sec. 3.1, Version 1.0 of ITS was obtained by integrating the eight Monte Carlo codes in Table II into a single card-image program library (SOURCE file). The resulting system is far more manageable for ourselves and for users alike than the older system involving separate program libraries for the individual codes. There is much common coding among the various member codes. Many geometry-independent routines such as scattering routines and utilities are identical in all of the codes. Even though the material geometry of the CYLTRAN codes is two dimensional, the tracking logic is fully three dimensional so that much of this logic is common to both the CYLTRAN and ACCEPT codes. The advantage of this common coding is that any programming modification, permanent or temporary, need only be done once instead of tediously repeating the modification for each individual code. Indeed, we have benefited from this advantage during the development of the ITS system when we added a common restart option and added the logic for photon transport down to 1.0 keV. These are just two of many examples.

The integration process also brought to the user a much more powerful system than was available with the older codes. CYLTRANP and the M codes represented new public releases of codes that had been developed for some time. Their expeditious release had been prevented by the burdensome task of preparing and maintaining multiple code packages. Even the maintenance of the unreleased versions for our own use had become quite difficult. Moreover, the ACCEPTP code was a new code that was simply a product of the integration process. We had wanted to develop this code for some time, but the effort required to do so as a separate entity was not justified in light of other priorities. On a smaller scale, but perhaps just as important, options in one code that should have been, but for one reason or another were not, added to the other codes simply became common coding as a result of the integration process.

4.2 Four Machines

To the maximum possible extent, the codes were rewritten in ANSI Standard Fortran 77. This does not mean, however, that the same Fortran version of a code is transportable -- i. e., that it will compile and load/link on more than one machine, or even on more than one operating system on a given machine. Software vendors can be counted on to implement nonstandard Fortran and to not implement standard Fortran in

their compilers. There are machine-dependent constants such as unit roundoff. The single precision word size of some machines may not provide sufficient accuracy. Necessary intrinsic routines may not be precisely the same, either in name or in function. Library functions and other utilities may not be generally available. Much effort was necessary to develop and test a system having full capability on CRAY, CDC, IBM, and VAX machines. Although this was done for Version 1.0, we ourselves have not been able to test subsequent versions on either CDC or IBM machines. Except for extensive testing of the IBM versions of all the codes by the Radiation Shielding Information Center, ORNL, we must now rely on user testing for these two vendors. Indeed, the ITS user community has now installed the full system on the machines of a great variety vendors, from main frames to work stations, and has run them under a number of operating systems. More recently, a more limited, but expanding, capability is being achieved on PCs.

The most serious obstacle to providing a multimachine capability in ITS had to do with word size. The original individual codes of the TIGER series were developed primarily on CDC machines having 60-bit words. Consequently, we encountered little difficulty in running the ITS member codes on a CRAY machine with its 64-bit word. Running on the VAX*, however, with its 32-bit word was not so simple. The difficulty stems primarily from the use of absolute tolerances in many of the member codes. Although we were able to run test problems by modifying certain tolerances, the modifications were problem dependent. Consequently, we made the decision not to support ITS application at less than a 60-bit word size.

Nevertheless, in order to make ITS available to users of machines with smaller word sizes, the SOURCE file is constructed such that double precision versions of the member codes can be obtained. Double precision versions are obtained whenever either of the update directives, *DEFINE,VAX or *DEFINE,IBM, are used in a correction run for either the XGEN or the ITS file (when the machine and code are selected) as discussed in Sec. 2.3.1. These versions were implemented by double precisioning all real constants and variables. To facilitate double precisioning of literal real constants, these constants were replaced throughout the file by real parameters beginning with the letter C. These real parameters are defined near the top of the file immediately after the specification of the integer parameters.

* Many institutions that heretofore found the cost of purchasing time on large scientific computers in order to run Monte Carlo radiation transport codes to be prohibitive are now finding it economical to do so on their own super-minicomputers and workstations. Even though these machines are slower than supercomputers, they often sit idle on nights and weekends.

We were frustrated in our initial attempts at constructing a double precision version of the ACCEPT code. In the original combinatorial logic of that code,^{8,9} mixed real and integer data are close packed in a single one-dimensional real array. This was tantamount to equivalencing reals with integers and was not compatible with our scheme for obtaining double precision. It was necessary to rewrite a substantial portion of the combinatorial geometry logic in order to eliminate this equivalencing. An unexpected benefit of this effort was simpler and more readable logic, and a reduction in run time of more than 10%.

The double precision versions of the ITS member codes have been successfully tested on VAX and IBM machines.

Many of the other machine and operating system dependencies were easily taken care of in the SOURCE file through the *IF DEF,[keyword] - *ENDIF construct described in Sec. 3.1. Machine constants, special features or deficiencies in Fortran compilers, and problems with intrinsic routines can usually be handled in this way. Random number generators and timing routines are the most obvious examples of routines that vary from installation to installation. See Appendix N for a discussion of some of the problems presented by random number generators.

Math library routines and the plot utility presented special problems. In the case of the library routines, our solution was to include Fortran versions of the necessary routines as part of the SOURCE file. These math library routines were needed in the M codes. They were taken from the FXMATH library at Sandia National Laboratories and rewritten in Fortran 77. Double precision versions were included, and embellishments similar to those in other routines of the SOURCE file were added.

Installation dependencies were important considerations in our decision not to provide a plot capability for certain member codes of ITS. We have simply defined the appropriate external routines from which a user may call whatever local plot utility he wishes. There are several good commercial plot utilities. These plot utilities are not generally available, however, and they are usually more sophisticated than we require. As discussed in Appendix I, we need only the capability for linear X-Y plots. Consequently, our own choice has been to use a very simple utility called WEASEL.²⁶ WEASEL was developed at Sandia National Laboratories and requires very little memory; it is therefore especially suitable for running on machines with limited fast memory.

4.3 Free-Format Input

The new free-format input scheme is likely to have more practical impact on users of the older versions of the ITS member codes than any other modification. This scheme has many advantages over the older formatted scheme:

- (1) The most obvious advantage is the elimination of the need for tedious column counting or careful positioning when entering input data.
- (2) The free formatting of data is coupled to a small set of descriptive keywords. This makes the input data file much more eye readable; the user can see at a glance the meaning of a particular group of numerical values.
- (3) The primary keywords are order independent so that, for example, the user does not have to remember whether the angular description of the source must be input before its energy distribution, whether electron flux options must be selected before photon flux options, etc.
- (4) An attempt has been made to implement defaults to the maximum possible extent. For example, the user doesn't have to specify 1000 primary histories, or 10 batches or an electron source since these are all defaults. In fact, a vacuous input file will not cause an abort until tracking begins because the geometry specification is the only input that does not have a default. The point is that users only ask for what they want. This eliminates the tedious chore of carefully going through a long list of input instructions and turning off lots of switches and inserting default or dummy input data for options that are of no interest.
- (5) The use of keywords with their appropriate defaults has made possible in the ITS member codes the selection through input of options and features that could only be obtained via update (code modification) in the original unintegrated codes. Consequently, the new input scheme gives us the best of both worlds by simultaneously minimizing both input and update.
- (6) The logic for the input scheme includes extensive internal checking for inconsistencies and other errors in the input data. Every effort is made to trap errors at the earliest possible time during execution by terminating the run with an informative diagnostic in order to avoid the necessity of time consuming and expensive debugging and traceback.

4.4 Parameterization

One of the more useful features of Fortran 77 that has been implemented to the maximum possible extent is parameterization -- the feature that permits the setting of values of certain real and integer constants at compilation time. In particular, parameterization has simplified the redimensioning of arrays in order to minimize memory usage and has facilitated the double precision modifications.

Virtually every array of every common block and every dimension statement of the ITS file has been parameterized. On machines with fast memory limitations such as the CDC 7600 or the CYBER 76,

parameterization provides a simple and convenient means of customizing the memory requirements of a member code to a particular application. Furthermore, the code provides information to assist in this task. The routine output of each code includes a table comparing memory allocation with memory requirements for the particular problem being run. With this information as a guide, the update processor, UPEML, can be used to reduce parameters that are larger than necessary. For example, the user may redimension the code for the actual number of materials or zones required for a given problem by changing a single parameter. In the older versions of the codes, it was necessary to carefully search out and change each individual array that depended on those dimensions. To facilitate the modification of these integer parameters, they are defined and their values set at the beginning of the ITS file. Finally, even on machines with no fast memory limitation, the use of parameterization permits more efficient allocation of memory resources and can also lead to a reduction in CPU time. See, however, the caution in Sec. 2.4.

As discussed in Sec. 4.2, parameterization was also helpful in providing double precision versions of the codes for users of machines with single precision word sizes of less than 60 bits. Our objective was to use double precision for all real numbers. This was easy enough to do for real variables with the DOUBLE PRECISION (A-H, O-Z) statement. However, the literal constants remained. Parameterization provided the solution. Virtually every literal constant has been replaced by a parameter, usually with a descriptive name, beginning with the letter C such as CONE for 1.0. These constants were then automatically included in the DOUBLE PRECISION statement. The real parameters are also set near the top of the ITS file just after the integer parameters.

4.5 Embellishments

Users of the versions of the ITS member codes before integration may find it surprising that this integrated program file is not more compact. The reason for this is that while much duplication was eliminated through integration, much was added in the way of embellishments.

The coding is now much more readable. The more complicated GO TO logic of the older versions has been replaced by the BLOCK-IF structure of Fortran 77. A hierarchy of indentation is used for nested BLOCK-IF's and DO-LOOP's. In the interest of a top-down linear flow of the logic, we have frequently seen fit to duplicate coding. By comparison with the codes before integration, ITS is extensively commented. All subroutine calls are highlighted. For the most part, format statements were collected at the top of each subroutine in Version 1.0 of ITS. In Version 3.0, however, we have gone much more to in-line formats where those formats contain descriptive information, because such formats provide the same kind of information as comments. Each subroutine begins with a banner that defines its function, shows its linkage to other program modules, and gives its dates of origination and last modification. Finally, attention has been given to many other aesthetic details of the coding, the cumulative effect of which is to facilitate an understanding of the logical structure.

4.6 Additional Restructuring Subsequent to Version 1.0

Of special note are some of the restructuring that occurred after Version 1.0. Much additional error checking was added. Subroutines INPUT and OUTPUT were each broken up into a number of smaller functional modules. The basic analytic-geometry routine of the combinatorial geometry of the three-dimensional ACCEPT codes (Subroutine GG) was completely rewritten with extensive comments. With this new software, we have consistently obtained run-time reductions of 10-15% for a variety of applications. Similarly, the module that provides enhanced simulation of atomic shell ionization and relaxation in the P codes (Subroutine PEAUF) was broken down and rewritten in modern Fortran 77, with much additional commenting, into a driver routine and a series of subroutines for each atomic shell. All integer toggles were replaced by logicals, and integer codes (e.g., primitive body types in the ACCEPT codes) were replaced by descriptive character strings.

5. Cross-Section Generating File - XGEN

XGEN combines the functions of the photon cross-section generating programs and the electron cross-section generating programs of the older unintegrated versions of the codes of the TIGER series into a single cross-section generating program. It is a card-image program library (SOURCE file) that uses the methods described in Sec. 3.1 to integrate the cross-section generating program for the standard codes with the cross-section generating program for the P codes for four machines. The integration process was performed in a fashion analogous to that employed to construct the card image program library for the Monte Carlo program file, ITS, as discussed in Secs. 4.1 and 4.2.

In the case of Version 1.0 of XGEN, this integration process was accomplished with a minimum of embellishments and little more than the Fortran changes that were absolutely necessary for Fortran 77 compilation. There are two notable exceptions to the latter. The formatted input used in all pre-ITS versions of the member codes has been replaced by a free-format scheme having the many advantages discussed in Sec. 4.3*. We also parameterized the number of materials for which cross-section data are to be generated in order to facilitate calculation for a larger number than the default value of five.

In Version 3.0, on the other hand, we have restructured XGEN much more extensively along the lines of the Monte Carlo program file as discussed in Sec. 4. This restructuring is still not as complete as we would like, but is a major improvement over Version 1.0. We carried out a line by line rewrite, converting the Fortran to the BLOCK-IF structure of the Fortran 77 standard, and a hierarchy of indentation has been used for nested BLOCK-IFs and DO-LOOPS. Changes to the names of common variables across modules were eliminated, so that only a single literal copy of each common block, or COMDECK in the parlance of the UPEML processor (see Sec. 3.1.1), appears in the XGEN file. Many more arrays have been parameterized, and error checking is much more extensive. Of special note is the parameterization of the length of the electron cross-section energy grid written to the Monte Carlo cross-section input file. As discussed under the CUTOFFS keyword of Sec. 2.3.2, this greatly facilitates the generation and use of electron cross sections that span an arbitrary energy range between 1.0 GeV and 1.0 keV. We have also parameterized literal real constants in a fashion similar to that described in Sec. 4.4 for the ITS file in order to facilitate the use of double precision when either of the directives, *DEFINE,VAX or *DEFINE,IBM, are used in an UPEML correction run.

* The maximum use of defaults in the cross-section input is emphasized by the fact that if no data is input, cross sections for 1.0 MeV in aluminum will be generated.

In addition to the input data discussed in Sec. 2.2, the executing codes read input data from the single data file, XDATA, described in Sec. 6. This combining of both the cross-section generating programs for electrons and photons and the corresponding data files greatly simplifies both the Job Control Language and the logic for manipulating the files within the programs.

There have been substantial improvements and changes to the particle cross sections between Versions 1.0 and 3.0. These are discussed in more detail in Appendix M, but will be mentioned briefly here. In Version 1.0, the photon cross sections were derived primarily from the compilations of Biggs and Lighthill,^{27,28} along with the tabulated pair production cross sections of Hubbell *et al.*²⁹ above 100.0 MeV. In Version 3.0, we have instead implemented the latest data base from the National Institute of Standards and Technology (see Sec. M.4 of Appendix M). The advantage of the latter cross sections is that they allow us to include, for the first time, coherent scattering in the simulation of photon transport and binding effects for both coherent and incoherent scattering (see Secs. M.5 and M.6 of Appendix M). Photon cross sections are generated from 1.0 GeV down to 1.0 keV. For the standard codes the K-fluorescence energies are obtained using the binding energies of Carlson³⁰; again, fluorescence and Auger production are allowed only for the highest-Z element of each material, regardless of its weight fraction. For the P codes, the photoionization data and relaxation data are taken from the compilation used in the SANDYL code.^{10,12}

The electron cross sections are also available from 1.0 GeV down to 1.0 keV, but are only generated up to the energy specified by the user via input. These cross sections are obtained primarily from a modified version of the DATAPAC program of the ETRAN Monte Carlo code system¹ with the following significant differences. The numerical electron elastic scattering cross sections of Haggmark *et al.*³¹ are used below 256.0 keV. The most important improvements in the physics of Version 3.0 relative to Version 1.0 are the new bremsstrahlung spectra and the improvements in the Landau/Blunck-Leisegang theory of collisional straggling. These are discussed in Secs. M.1 and M.3, respectively, of Appendix M. There are also other changes of lesser importance. In Version 1.0, the density effect correction to the electronic stopping power is computed using the scheme described by Sternheimer and Peierls.³² A more fundamental approach is employed in Version 3.0 as part of the implementation of more recent collisional stopping powers (see Sec. M.2 of Appendix M). For the standard codes, K-shell electron impact ionization cross sections are computed according to the theory of Kolbenedt.³³ On the other hand, for the P codes the electron impact ionization cross sections for all shells having binding energies greater than 1.0 keV are obtained using the theory of Gryzinski.³⁴

6. Cross-Section Data File - XDATA

In Version 1.0, XDATA was obtained by combining and improving the two data files that were used by the two separate cross-section generating programs in the older unintegrated versions of the ITS member codes. The photon data were essentially the cross-section data of Biggs and Lighthill^{27,28} in modified format. The electron data consisted primarily of LIBRARY TAPE 2 of the ETRAN Monte Carlo code system.¹ This tape is distinguished from other library tapes of the ETRAN system in that the empirical corrections to the bremsstrahlung cross sections were based upon the experimental data of Rester³⁵ and Aiginger.³⁶ To these basic data we added (a) the electron elastic scattering data of Haggmark *et al.*,³¹ (b) the high-energy pair production data of Hubbell *et al.*,²⁹ and (c) the ionization/relaxation data of the SANDYL code.^{10,12} Combining the two data files led to a simplification of both the Job Control Language and the internal logic of the cross-section generating programs.

In Version 3.0 we have retained convenience of the single-file approach, but this file is now much larger and much of the older data has been replaced. The increased size is due primarily to (a) the data that replaces the photon cross-section data of Biggs and Lighthill, (b) the new numerical bremsstrahlung cross sections, (c) the expanded data required for the new algorithms for sampling from the Landau/Blunck-Leisegang energy loss straggling distributions, and (d) the data required by the new algorithm for generating the density-effect correction to the electron stopping powers. These data are a consequence of the improved physical model discussed briefly in Sec. 5, and in more detail in Appendix M.

7. Concluding Remarks

Version 1.0 of the ITS system represented the culmination of more than a decade and a half of effort involving contributions by many different people. Over the last seven years Versions 1.0, 2.0, and 2.1 have met with considerable success in a great variety of applications by a world-wide user community. Nevertheless, we never expected and it was never intended that this software should remain static. Version 3.0 represents the first major upgrade of the system and the extensive improvements are as much a consequence of the many valuable suggestions by the user community as they are the results of our own limited experience. Even now, though, we are contemplating a number of extensions and improvements to the cross sections, the Monte Carlo model, and the software structure of Version 3.0. In this regard, we continue to solicit feedback from the user community on further modifications that will enhance the utility and accuracy of the system. It is our intention to make subsequent versions of the system available in as timely a fashion as possible.

We sincerely hope that Version 3.0 of the ITS system will prove as successful a tool as Version 1.0 in the design and analysis of that broad class of science and engineering applications in which the transport of electron, x-ray, and gamma-ray radiation plays an important role.

Appendix A: Installation

A.1 Overview

As noted in the introduction, there are four components in the ITS package:

- 1) XDATA --- The electron/photon cross-section data file
- 2) XGEN --- The cross-section generating program SOURCE file
- 3) ITS --- The Monte Carlo program SOURCE file
- 4) UPEML --- A file containing the update processor²⁵ which emulates the standard CDC UPDATE processor.²⁴

We assume that these files have been provided on a magnetic tape. In this appendix we give step-by-step instructions for the one time only installation of the ITS package.*

A.2 Step-by-Step Installation Guide

STEP 1. Copy the four files named in A.1 from the tape to the storage system on your computer. Note that the cross-section data file, XDATA, is comprised of 96 column records while the remaining three files are 80 column records.

STEP 2. After choosing the appropriate machine-dependent source code, compile and link the UPDATE emulator. Note that you might need to modify the OPEN statement for the file "oldpl" in subroutine FOPEN in the UPEML package as noted in the comment cards found in this subroutine. This should represent the only machine dependency in UPEML. Retain the executable of this program; it is required for both this installation process ("creation" run) and for subsequent code selections and modifications as described in Secs. 2.2.1, 2.3.1, and 3.3 ("correction" runs). (NOTE that you can skip this step if your system already has an update processor that recognizes the standard CDC conventions.)

* Although the Job Control Language for accomplishing the creation runs will very much depend on the local operating system, we provide some guidance for setting up the job control stream in Appendix K. This appendix also includes a discussion of all input/output units.

STEP 3. Create OLDPLs (old program libraries) from both the XGEN and ITS SOURCE files using UPEML. The procedure is to run the UPEML executable created in Step 2 with the following input*:

```
{eor}
N,L
"source cards from tape"
...
{eor}
```

This creation run option shows the use of the "N" option to generate a new program library (NEWPL) and also a listing of all the card images included in this NEWPL along with their deck name (or common deck name) and sequence number identifiers (this is requested by the "L"). The command line is then followed by the SOURCE cards from tape appended as in-line input to the program (i.e., Fortran unit 5 input). Alternatively, the source cards can be placed into a file, and UPEML can be instructed to read the card images from that file rather than in line. This is accomplished by adding an "I" (comma delimited) to the command line. Then UPEML will read the file "srcdk" for the source input. The user must assign the file he desires to this file name in the Job Control Language. That is, the Job Control Language should equivalence the SOURCE file, XGEN or ITS, to the default file name, "srcdk". Alternatively, adding "I=[filename]" to the command line will override "srcdk" with [filename].

The UPEML processor will indicate that this is a creation run.

The processor will create a binary NEWPL on the file "newpl". Note that future correction runs will read this NEWPL as an existing OLDPL on the file "oldpl". The binary program library file created from the SOURCE file should be conveniently named and saved as a permanent disk file. The processor will also create a listing of the program library on the file "listfl". This listing should always be printed for future reference; it will be needed to provide deck names (or common deck names) and sequence numbers when using the *INSERT, *DELETE, and *BEFORE directives in future correction runs as described in Sec. 3.3.

Run UPEML twice: once on the XGEN SOURCE file and once using the ITS SOURCE file as the input. The two resulting program libraries should not be changed.

* More detailed information on the use of the UPEML processor to make these "creation" runs can be found in Sec. 4 and Ref. 25.

WARNING - to facilitate communication with the authors concerning bugs or desired program modifications, make all local modifications to the program libraries by using the UPEML processor. That is, make all modifications as UPEML "correction runs" as discussed in Section 3.3. DO NOT use a text editor to make local modifications to either the program libraries or the SOURCE files!!!

When you have successfully performed Steps 1, 2, and 3, the installation of the ITS package is complete. Now proceed to Sec. 2 of this document, which describes how the member codes of ITS may be run using the libraries created in this installation.

Appendix B: Sample Input Streams

In this Appendix we show sample input streams for running each of the eight member codes of ITS. These examples are set up to sequentially accomplish within a single job the four primary tasks:

- (1) Select a cross-section generating code.
- (2) Execute the cross-section generating code.
- (3) Select a Monte Carlo code.
- (4) Execute the Monte Carlo code.

The portions of the input streams required for these tasks are separated from one another by {eor}s. Users may find it more efficient to break this single job down into multiple jobs that perform a smaller number of tasks. For example, tasks (1) and (2) can be run in one job and the cross-section output saved for repeated use by the Monte Carlo codes. If the codes are not being modified from one run to the next, the binary executables can be saved, and it will not be necessary to repeat steps (1) and (3). Nevertheless, beginning users will find it easier to accomplish all four tasks within each job using input streams like those given in the following examples.

The reader should note the liberal use of the commenting feature (asterisk in column 1) of the free format execution input [tasks (2) and (4)] in the following examples. These are useful for inserting additional descriptive information into these input streams or for deactivating keyword inputs instead of deleting them. Such comments are permitted anywhere in the input stream where they will be processed by the external parsing routine OPREAD. They are not allowed where they would be read directly by Fortran READ statements instead of calling OPREAD. For example, they cannot be used immediately after a line containing the keywords, TITLE, SPECTRUM, RANDOM-NUMBER, and USER. In addition for the ACCEPT codes, they cannot be used in the middle of the specification for a given input body or input zone, or after the END keyword that terminates input-zone input when the volumes of the subzones are to be read in (see GEOMETRY keyword).

Also, explicit execution keywords are often used in the execution input where they are not required (i.e., they specify default values); this merely makes the input streams more readable and explicitly shows all options in effect for the run. ENERGY, ELECTRONS, DIRECTION, and BATCHES are examples of keywords that are often used to explicitly define default inputs in the sample input streams that follow.

The lines beginning with */ followed by at least one space are comments that are ignored by the UPDATE processor when processing input for tasks (1) and (3). Such comments must not appear within or immediately before a sequence of Fortran statements that is being inserted; rather, they should appear at the end of such a sequence.

In the following examples, the command line, "F,P" appears explicitly in the inputs for tasks (1) and (3). This assumes that the input files for these tasks are being read from the default input device as described in Sec. 2.1.1. If the "I" parameter were included, the command line "F,P,I" would be the only input from the default input device (usually Fortran unit 5), and the remaining inputs for tasks (1) and (3) would have to be assigned to file "srcdk" (see Sec. 2.1.1).

In a number of the sample input streams, examples of Fortran 77 PARAMETER modification are included as part of the input for tasks (1) or (3). These are usually optional, but may be required when running complex problems on machines with limited fast memory. Moreover, they may significantly reduce computing costs by customizing memory allocation to a particular problem. Note also the use of "*D," and "*I," as abbreviations for "*DELETE," and "*INSERT," respectively. In rare cases where user requirements exceed default allocations, the run will terminate with a message informing the user as to which default Fortran PARAMETER must be increased.

In addition to modifications to the Fortran that the more sophisticated user may wish to add to the input for tasks (1) and (3), there are a couple of modifications that may be required. As discussed in Sec. 2.3.1, when running either of the M codes, Subroutine BFLD must be modified to return values of the six components of the externally applied electric and magnetic fields at any point within any input zone for which a non-zero field has been specified (see the GEOMETRY keyword in Sec. 2.3.2 for CYLTRANM and ACCEPTM). Examples of such modifications are shown in the inputs for task (3) in Secs. B.7 and B.8. The other much more rare case where modifications are required occurs when the default electron energy grid of the cross-section generator will not span the desired energy range for the Monte Carlo run -- i. e., from the maximum source energy to the desired global electron cutoff energy. In this case the Fortran PARAMETERS INX and INMAX in the cross-section generator and Monte Carlo codes, respectively, must be modified as described under the description of the CUTOFFS keyword of Sec. 2.3.2 in order to make possible the generation and use the expanded electron energy grid. An example of these modifications are shown in Sec. B.6 in the input for tasks (1) and (3). However, they are commented out in that example since the span of the default energy grid is adequate for the Monte Carlo run of that example.

These examples can only be properly understood after reading Sec. 2.

B.1 Sample Input for Running TIGER Code

```
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
{eor}
ECHO 1
MATERIAL TA
MATERIAL AL
TITLE
  1.0 MEV STANDARD CODES CROSS SECTIONS FOR TA AND AL
ENERGY 1.0
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,TIGER
*DEFINE,RNG1
*/ VAX AND IBM ARE AUTOMATICALLY DOUBLE PRECISIONED AS OF VERSION 3.0
{eor}
ECHO 1
TITLE
...1.0 MEV TA/AL TEST PROBLEM
***** GEOMETRY *****
* MATERIAL SUBZONES THICKNESS ELECTRON-CUTOFF FORCING
GEOMETRY 2
      1          2          0.007
      2          2          0.05
***** SOURCE *****
ELECTRONS
ENERGY 1.0
CUTOFFS 0.05 0.001
* DEFAULT DIRECTION
DIRECTION 0.0
***** OUTPUT OPTIONS *****
ELECTRON-ESCAPE
  NBINE 2
  NBINT 4
PHOTON-ESCAPE
  NBINE 2
  NBINT 4
ELECTRON-FLUX 1 2
  NBINE 2
  NBINT 2
PHOTON-FLUX 1 2
  NBINE 2
  NBINT 2
***** OTHER OPTIONS *****
HISTORIES 10000
BATCHES 10
* ... X-RAY PRODUCTION SCALING
SCALE-BREMS 500.
* SCALE-BREMS 5.0 2
```

B.2 Sample Input for Running CYLTRAN Code

```
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
{eor}
MATERIAL TA
MATERIAL AL
TITLE
  1.0 MEV STANDARD CODES CROSS SECTIONS FOR TA AND AL
ENERGY 1.0
*ENERGY 2.0
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,CYLTRAN
*DEFINE,RNG1
{eor}
ECHO 1
TITLE
  ...1.0 MEV TA/AL TEST PROBLEM
***** GEOMETRY *****
* ZMIN ZMAX RMIN RMAX MATERIAL NPHI NRHO NZ CUTOFF FORCING
GEOMETRY 3
-0.007 0.0 0.0 10.0 1 1 2 2
  0.1 0.15 0.0 10.0 2 1 2 2
  0.0 0.1 0.0 10.0
***** SOURCE *****
ELECTRONS
ENERGY 1.0
CUTOFFS 0.05 0.001
POSITION 0.0 0.0 -0.007
  RADIUS 2.50
* DEFAULT DIRECTION
DIRECTION 0.0 0.0
***** OUTPUT OPTIONS *****
ELECTRON-ESCAPE
  NBINE 2
  NBINT 4
PHOTON-ESCAPE
  NBINE 2
  NBINT 4
ELECTRON-FLUX 1 2
  NBINE 2
  NBINT 2
PHOTON-FLUX 1 2
  NBINE 2
  NBINT 2
***** OTHER OPTIONS *****
HISTORIES 10000
BATCHES 10
* ... X-RAY PRODUCTION SCALING
SCALE-BREMS 500.
```

B.3 Sample Input for Running ACCEPT Code

```
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
{eor}
MATERIAL TA
MATERIAL AL
TITLE
  1.0 MEV STANDARD CODES CROSS SECTIONS FOR TA AND AL
ENERGY 1.0
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,ACCEPT
*/ DEFINE,PLOTS
*DEFINE,RNG1
{eor}
ECHO 1
TITLE
  ...1.0 MEV TA/AL TEST PROBLEM
***** GEOMETRY *****
GEOMETRY
  RCC      0.00      0.00      0.00      0.00      0.00      -0.007
           10.0
  RCC      0.00      0.00      0.00      0.00      0.00      0.10
           10.0
  RCC      0.00      0.00      0.10      0.00      0.00      0.05
           10.0
  SPH      0.00      0.00      0.00      12.0
  END
* TANTALUM
  ZSUB1  +1
  1 2 2
* ALUMINUM
  ZSUB2  +3
  1 2 2
* VOID
  Z3    +2
* ESCAPE
  Z4   +4   -1   -2   -3
  END
* MATERIAL CUTOFF STRETCHING
  1
  2
  0
  0
***** SOURCE *****
ELECTRONS
ENERGY 1.0
CUTOFFS 0.05 0.001
```

```

POSITION 0.0 0.0 -0.007
RADIUS 2.50
* DEFAULT DIRECTION
DIRECTION 0.0 0.0
***** OUTPUT OPTIONS *****
ELECTRON-ESCAPE
  NBINE 2
  NBINT 4
PHOTON-ESCAPE
  NBINE 2
  NBINT 4
ELECTRON-FLUX 1 2
  NBINE 2
  NBINT 2
PHOTON-FLUX 1 2
  NBINE 2
  NBINT 2
***** OTHER OPTIONS *****
HISTORIES 10000
BATCHES 10
* ... X-RAY PRODUCTION SCALING
SCALE-BREMS 500.
* PLOTS 3
* -15.0 15.0 -15.0 15.0 180. 90.
* -15.0 15.0 -15.0 15.0 0.0 0.0
* -15.0 15.0 -15.0 15.0 180. 30.

```

B.4 Sample Input for Running TIGERP Code

```
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,PCODES
*IDENT,PMOD
*/ THE FOLLOWING EXAMPLES OF PARAMETER MODIFICATION ARE USUALLY
*/ OPTIONAL, BUT MAY SIGNIFICANTLY REDUCE COMPUTER COSTS BY REDUCING
*/ MEMORY REQUIREMENTS.
*D,PARAMS.9
    PARAMETER ( INMT=2,      INEM=1)
{eor}
MATERIAL TA
MATERIAL AL
TITLE
    1.0 MEV P-CODES CROSS SECTIONS FOR TA AND AL
ENERGY 1.0
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,TIGER
*DEFINE,PCODES
*DEFINE,RNG1
{eor}
ECHO 1
TITLE
...1.0 MEV TA/AL TEST PROBLEM
***** GEOMETRY *****
* MATERIAL SUBZONES THICKNESS ELECTRON-CUTOFF FORCING
GEOMETRY 2
    1 2 0.007
    2 2 0.05
***** SOURCE *****
ELECTRONS
* PHOTONS
ENERGY 1.0
CUTOFFS 0.05 0.001
* DEFAULT DIRECTION
DIRECTION 0.0
***** OUTPUT OPTIONS *****
ELECTRON-ESCAPE
    NBINE 2
    NBINT 4
PHOTON-ESCAPE
    NBINE 2
    NBINT 4
ELECTRON-FLUX 1 2
    NBINE 2
    NBINT 2
```

PHOTON-FLUX 1 2
NBINE 2
NBINT 2
***** OTHER OPTIONS *****
HISTORIES 10000
BATCHES 10
* ... X-RAY PRODUCTION SCALING
SCALE-BREMS 500.

B.5 Sample Input for Running CYLTRANP Code

```
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,PCODES
*IDENT,PMOD
*/ THE FOLLOWING EXAMPLES OF PARAMETER MODIFICATION ARE USUALLY
*/ OPTIONAL, BUT MAY SIGNIFICANTLY REDUCE COMPUTER COSTS BY REDUCING
*/ MEMORY REQUIREMENTS.
*D,PARAMS.9
      PARAMETER ( INMT=2,      INEM=1)
{eor}
MATERIAL TA
MATERIAL AL
TITLE
  1.0 MEV P-CODES CROSS SECTIONS FOR TA AND AL
ENERGY 1.0
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,CYLTRAN
*DEFINE,PCODES
*DEFINE,RNG1
{eor}
ECHO 1
TITLE
  ...1.0 MEV TA/AL TEST PROBLEM
***** GEOMETRY *****
* ZMIN ZMAX RMIN RMAX MATERIAL NPHI NRHO NZ CUTOFF FORCING
GEOMETRY 3
  -0.007      0.0      0.0      10.0      1  1  2  2
  0.1      0.15      0.0      10.0      2  1  2  2
  0.0      0.1      0.0      10.0
***** SOURCE *****
ELECTRONS
ENERGY 1.0
* ENERGY 2.0
CUTOFFS 0.05 0.001
POSITION 0.0 0.0 -0.007
  RADIUS 2.50
* DEFAULT DIRECTION
DIRECTION 0.0 0.0
***** OUTPUT OPTIONS *****
ELECTRON-ESCAPE
  NBINE 2
  NBINT 4
PHOTON-ESCAPE
  NBINE 2
  NBINT 4
```

ELECTRON-FLUX 1 2
NBINE 2
NBINT 2
PHOTON-FLUX 1 2
NBINE 2
NBINT 2
***** OTHER OPTIONS *****
HISTORIES 10000
BATCHES 10
* ... X-RAY PRODUCTION SCALING
SCALE-BREMS 500.

B.6 Sample Input for Running ACCEPTP Code

```
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,PCODES
*IDENT,PMOD
*/ THE FOLLOWING UPDATE OF INX IS NEEDED TO GENERATE A CROSS-SECTION SET
*/ WITH EXPANDED ELECTRON ENERGY GRID. THE PARAMETER INMAX IN ANY MONTE
*/ CARLO RUN THAT USES THE RESULTING CROSS-SECTION SET MUST BE
*/ EQUAL TO OR GREATER THAN THE VALUE OF NMAX TO BE FOUND IN THE UNIT
*/ IOUT (DEFAULT IOUT=6) OUTPUT FILE OF THE CROSS-SECTION GENERATOR RUN
*/ (SETTING INMAX OF THE MONTE CARLO RUN TO THE VALUE OF INX IS ALWAYS
*/ SUFFICIENT). THE EXPANDED ELECTRON ENERGY GRID IS NOT NEEDED IN
*/ MONTE CARLO SAMPLE PROBLEMS WHERE THE SOURCE ELECTRON ENERGY IS 1.0
*/ MeV AND THE GLOBAL ELECTRON CUTOFF ENERGY IS 0.05 MeV.
*/ *D,PARAMS.5
*/ PARAMETER ( INX=88,   INX1=INX+1,   INL=501)
*/
*/ THE FOLLOWING EXAMPLES OF PARAMETER MODIFICATION ARE USUALLY
*/ OPTIONAL, BUT MAY SIGNIFICANTLY REDUCE COMPUTER COSTS BY REDUCING
*/ MEMORY REQUIREMENTS.
*/D,PARAMS.9
      PARAMETER ( INMT=2,   INEM=1)
{eor}
ECHO 1
MATERIAL TA
MATERIAL AL
TITLE
  1.0 MEV P-CODES CROSS SECTIONS FOR TA AND AL
ENERGY 1.0
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,ACCEPT
*DEFINE,PCODES
*/ DEFINE,PLOTS
*DEFINE,RNG1
*IDENT,PMOD
*/ INMAX UPDATE FOR USING CROSS-SECTION SET WITH EXPANDED ELECTRON
*/ ENERGY GRID: THE PARAMETER INMAX IN THE MONTE CARLO CODE MUST BE
*/ EQUAL TO OR GREATER THAN THE VALUE OF NMAX IN THE UNIT IOUT OUTPUT
*/ OF THE CROSS-SECTION GENERATOR (SETTING INMAX OF THE MONTE CARLO
*/ RUN TO THE VALUE OF INX IS ALWAYS SUFFICIENT). EXPANDED ELECTRON
*/ ENERGY GRID IS NOT NEEDED IN THIS EXAMPLE WHERE THE SOURCE ELECTRON
*/ ENERGY IS 1.0 MeV AND THE GLOBAL ELECTRON CUTOFF ENERGY IS 0.05 MeV.
*/ PARAMETER ( INMT=2,   INEM=1,   INMAX=88,   NSURV=2775,
*/
```

```

*/ THE FOLLOWING EXAMPLES OF PARAMETER MODIFICATION ARE USUALLY
*/ OPTIONAL, BUT MAY SIGNIFICANTLY REDUCE COMPUTER COSTS BY REDUCING
*/ MEMORY REQUIREMENTS.
*D, PARAMS. 9
    PARAMETER ( INMT=2,      INEM=1,      INMAX=64,      NSURV=2775,
*D, PARAMS. 11, PARAMS. 14
    $          INRANG=34,    INTANG=INMAX/4+1, INEEL=13,    INPEL=21,
    $          INEPS=9,      INGAS=1000,    INLAN=5000, INPPS=21,
    $          INLAMB=1591,   JAHSUB=51,    IJSPEC=1,   JATPR=698,
    $          JATAN=799,    INTAB=10,    IMTAX=63 )
*D, PARAMS. 28, PARAMS. 30
    PARAMETER ( IKMAX = 4,    IJMAX = 2,
    $          IKPMAX = 4,    IJPMAX = 2,
    $          INIZON = 4,    INSZON = 9)
*D, PARAMS. 37
    PARAMETER ( IJSMAX = 1, IJFMAX =2,
*D, PARAMS. 39, PARAMS. 42
    $          IJFMXP = 2,      IJFMP1 = IJFMXP+1,
    $          IKFMAX = 2,      IKFMP1 = 2,
    $          IKFMX1 = IKFMAX+1, IKFMP1 = IKFMXP+1,
    $          INLF = 8,        INLFP = 8)
*D, PARAMS. 81, PARAMS. 82
    $ IFPD = 43,    INUMR = 4,    ITMA = 88,    NAZ = 5,
    $ IJTY = 4,    IARB = 1,    NIEWS = 3,    NCZONE = 20,
{eor}
ECHO 1
TITLE
...1.0 MEV TA/AL TEST PROBLEM
***** GEOMETRY *****
GEOMETRY
RCC      0.00      0.00      0.00      0.00      0.00      -0.007
          10.0
RCC      0.00      0.00      0.00      0.00      0.00      0.10
          10.0
RCC      0.00      0.00      0.10      0.00      0.00      0.05
          10.0
SPH      0.00      0.00      0.00      12.0
END
ZSUB001  +1
1 2 2
ZSUB002  +3
1 2 2
Z003    +2
Z004    +4   -1   -2   -3
END
* MATERIAL CUTOFF STRETCHING
1
2
0
0

```

```

***** SOURCE *****
ELECTRONS
ENERGY 1.0
CUTOFFS 0.05 0.001
*CUTOFFS 0.10 0.001
POSITION 0.0 0.0 -0.007
  RADIUS 2.50
* DEFAULT DIRECTION
DIRECTION 0.0 0.0
***** OUTPUT OPTIONS *****
ELECTRON-ESCAPE
  NBINE 2
  NBINT 4
PHOTON-ESCAPE
  NBINE 2
  NBINT 4
ELECTRON-FLUX 1 2
  NBINE 2
  NBINT 2
PHOTON-FLUX 1 2
  NBINE 2
  NBINT 2
***** OTHER OPTIONS *****
HISTORIES 10000
BATCHES 10
* ... X-RAY PRODUCTION SCALING
SCALE-BREMS 500.
* PLOTS 3
* -15.0 15.0 -15.0 15.0 180. 90.
* -15.0 15.0 -15.0 15.0 0.0 0.0
* -15.0 15.0 -15.0 15.0 180. 30.

```

B.7 Sample Input for Running CYLTRANM Code

```
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
{eor}
MATERIAL TA
MATERIAL AL
TITLE
  1.0 MEV STANDARD CODES CROSS SECTIONS FOR TA AND AL
ENERGY 1.0
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,CYLTRAN
*DEFINE,MCODES
*/ DEFINE,PLOTS
*DEFINE,RNG1
*IDENT,TEST1
*/ THE FOLLOWING UPDATE RESULTS IN A ZERO ELECTRIC FIELD AND A MAGNETIC
*/ FIELD OF 10 KILOGAUSS IN THE AXIAL DIRECTION FOR ANY INPUT ZONE FOR
*/ WHICH A NONZERO FIELD IS SPECIFIED.
*I,BFLD.60
    EZ = CZERO
    BX = CZERO
    BY = CZERO
    BZ = CONE
    RETURN
*/
*/ THE FOLLOWING EXAMPLES OF PARAMETER MODIFICATION ARE USUALLY
*/ OPTIONAL, BUT MAY SIGNIFICANTLY REDUCE COMPUTER COSTS BY REDUCING
*/ MEMORY REQUIREMENTS.
*D,PARAMS.9
    PARAMETER ( INMT=2,      INEM=1,      INMAX=64,      NSURV=2775,
*D,PARAMS.11,PARAMS.14
    $          INRANG=34,    INTANG=INMAX/4+1, INEEL=13,    INPEL=21,
    $          INEPS=9,      INGAS=1000,    INLAN=5000, INPPS=21,
    $          INLAMB=1591,  JAHSUB=51,    IJSPEC=1,   JATPR=698,
    $          JATAN=799,    INTAB=10,     IMTAX=63 )
*D,PARAMS.28,PARAMS.30
    PARAMETER ( IKMAX = 4,    IJMAX = 2,
    $          IKPMAX = 4,    IJPMAX = 2,
    $          INIZON = 3,    INSZON = 9)
*D,PARAMS.37
    PARAMETER ( IJSMAX = 1,  IJFMAX =2,
*D,PARAMS.39,PARAMS.42
    $          IJFMP1 = IJFMP1+1,
    $          IKFMP1 = 2,
    $          IKFMP1 = IKFMP1+1,  IKFMP1 = IKFMP1+1,
    $          INLFP = 8,          INLFP = 8)
```

```

(eor)
ECHO 1
TITLE
...1.0 MEV TA/AL TEST PROBLEM
***** GEOMETRY *****
* ZMIN ZMAX RMIN RMAX MATERIAL NPHI NRHO NZ FIELD CUTOFF FORCING
GEOMETRY 3
-0.007 0.0 0.0 10.0 1 1 2 2
5.0 5.05 0.0 10.0 2 1 2 2
0.0 5.0 0.0 10.0 0 0 0 0 1
* -0.007 0.0 0.0 10.0 1 1 1 4
***** SOURCE *****
ELECTRONS
ENERGY 1.0
CUTOFFS 0.05 0.001
POSITION 0.0 0.0 -0.007
RADIUS 2.50
* DEFAULT DIRECTION
DIRECTION 0.0 0.0
***** OUTPUT OPTIONS *****
ELECTRON-ESCAPE
NBINE 2
NBINT 4
PHOTON-ESCAPE
NBINE 2
NBINT 4
ELECTRON-FLUX 1 2
NBINE 2
NBINT 2
PHOTON-FLUX 1 2
NBINE 2
NBINT 2
***** OTHER OPTIONS *****
HISTORIES 10000
BATCHES 10
* ... X-RAY PRODUCTION SCALING
SCALE-BREMS 500.
*PLOTS 0.0 10.0 -2.0 8.0
* ORBITS 100

```

B.8 Sample Input for Running ACCEPTM Code

```
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
{eor}
MATERIAL TA
MATERIAL AL
TITLE
  1.0 MEV STANDARD CODES CROSS SECTIONS FOR TA AND AL
ENERGY 1.0
{eor}
F,P
*IDENT,DEFINE
*DEFINE,VAX
*DEFINE,ACCEPT
*DEFINE,MCODES
*/ DEFINE PLOTS
*DEFINE,RNG1
*IDENT,TESTA
*/ THE FOLLOWING UPDATE RESULTS IN A ZERO ELECTRIC FIELD AND A MAGNETIC
*/ FIELD OF 10 KILOGAUSS IN THE AXIAL DIRECTION FOR ANY INPUT ZONE FOR
*/ WHICH A NONZERO FIELD IS SPECIFIED.
*I,BFLD.60
    EZ = CZERO
    BX = CZERO
    BY = CZERO
    BZ = CONE
    RETURN
*/
*/ THE FOLLOWING EXAMPLES OF PARAMETER MODIFICATION ARE USUALLY
*/ OPTIONAL, BUT MAY SIGNIFICANTLY REDUCE COMPUTER COSTS BY REDUCING
*/ MEMORY REQUIREMENTS.
*D,PARAMS.9
    PARAMETER ( INMT=2,      INEM=1,      INMAX=64,      NSURV=2775,
*D,PARAMS.28,PARAMS.30
    PARAMETER ( IKMAX   = 4,      IJMAX   = 2,
    $             IKPMAX  = 4,      IJPMAX  = 2,
    $             INIZON  = 4,      INSZON  = 9)
*D,PARAMS.37
    PARAMETER ( IJSMAX = 1, IJFMAX =2,
*D,PARAMS.39,PARAMS.42
    $             IJFMXP = 2,          IJFMP1 = IJFMXP+1,
    $             IKFMAX = 2,          IKFMXP = 2,
    $             IKFMX1 = IKFMAX+1,   IKFMP1 = IKFMXP+1,
    $             INLF   = 8,          INLFP  = 8)
*D,PARAMS.81,PARAMS.82
    $ IFPD = 43,   INUMR = 4,   ITMA  = 88,   NAZ   = 5,
    $ IJTY = 4,   IARB  = 1,   NVIEW = 3,   NCZONE = 20,
{eor}
ECHO 1
TITLE
...1.0 MEV TA/AL TEST PROBLEM
```

```

***** GEOMETRY *****
GEOMETRY
  RCC      0.00      0.00      0.00      0.00      0.00      -0.007
           10.0
  RCC      0.00      0.00      0.00      0.00      0.00      5.00
           10.0
  RCC      0.00      0.00      5.00      0.00      0.00      0.05
           10.0
  SPH      0.00      0.00      0.00      12.0
  END
  ZSUBX1   +1
  1 2 2
  ZSUBX2   +3
  1 2 2
  ZX3     +2
  ZX4     +4   -1   -2   -3
  END
* MATERIAL FIELD CUTOFF STRETCHING
  1      0
  2      0
  0      1
  0      0
***** SOURCE *****
ELECTRONS
ENERGY 1.0
CUTOFFS 0.05 0.001
POSITION 0.0 0.0 -0.007
  RADIUS 2.50
* DEFAULT DIRECTION
DIRECTION 0.0 0.0
***** OUTPUT OPTIONS *****
ELECTRON-ESCAPE
  NBINE 2
  NBINT 4
PHOTON-ESCAPE
  NBINE 2
  NBINT 4
ELECTRON-FLUX 1 2
  NBINE 2
  NBINT 2
PHOTON-FLUX 1 2
  NBINE 2
  NBINT 2
***** OTHER OPTIONS *****
HISTORIES 10000
BATCHES 10
* ... X-RAY PRODUCTION SCALING
SCALE-BREMS 500.
* PLOTS 3
* -15.0  15.0  -15.0  15.0  180.  90.
* -15.0  15.0  -15.0  15.0  0.0  0.0
* -15.0  15.0  -15.0  15.0  180.  30.

```

Appendix C: TIGER Codes - 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.

C.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 Appendix L), 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 as described in Sec. 2.2.2.

Other optional zone-dependent parameters are also available to the user. He may raise the electron cutoff energy above its global value (see keyword CUTOFFS), and he may force a certain fraction of photons entering a given input zone to interact (see Secs. H.4 and H.6 of Appendix H). Specification of these parameters is described under keyword GEOMETRY.

C.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. The fact that the default source is located at $z=0.0$ leads to a certain convention in defining particle escape. Particle escape is classified according to the following definitions:

- a) Transmitted radiation is radiation that escapes from the maximum-z boundary of the problem.
- b) Reflected radiation is radiation that escapes from the minimum-z boundary of the problem.

These definitions are maintained regardless of how the source is defined.

Appendix D: CYLTRAN Codes - 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 his correction run (see Sec. 3.3) 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 and PHOTON-ESCAPE) and azimuthally-dependent charge deposition, energy deposition, and particle fluxes (see keywords ELECTRON-FLUX, PHOTON-FLUX, and GEOMETRY, as well as the discussion of automatic subzoning in Appendix L).

D.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. D-1. 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. D-1) 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. D-1 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

as described in Sec. 2.2.2. 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 of Sec. 2.3.2. 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.

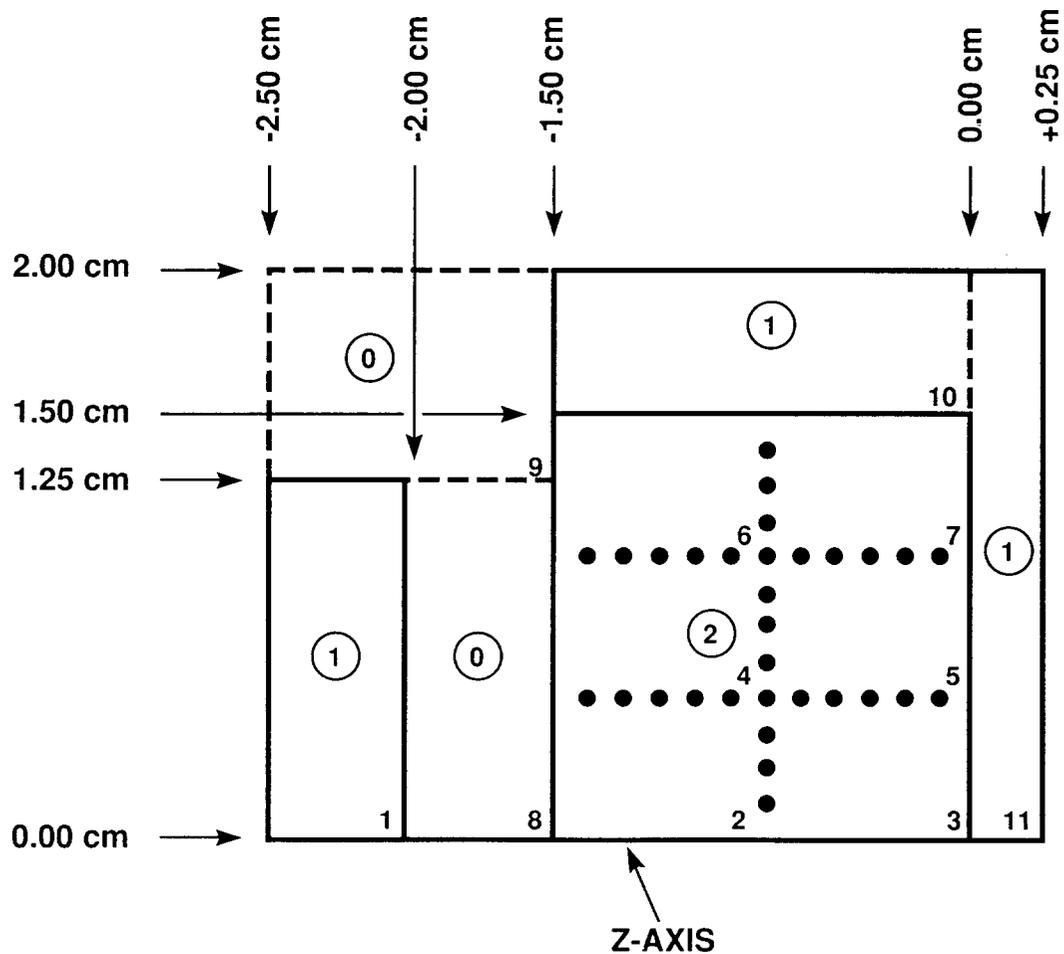


Figure D-1: Example of the Half Section of a Problem Cylinder

The following input cards describe the the problem geometry illustrated in Fig. D-1.

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

Other optional input-zone-dependent parameters are also available to the user. He may raise the electron cutoff energy above its global value (see keyword CUTOFFS), and he may force a certain fraction of photons entering a given input zone to interact (see Secs. H.4 and H.6 of Appendix H). Finally, in the case of CYLTRANM, he may specify the presence of macroscopic electric fields (in voids only) and/or magnetic fields. Specification of these parameters is also described under the GEOMETRY keyword.

D.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 default source is a disk source (beam) at the minimum-z boundary of the problem cylinder, a certain convention is used in defining particle escape. Particle escape is classified according to the following definitions:

- a) Transmitted radiation is radiation that escapes from the maximum-z boundary of the problem cylinder.
- b) Reflected radiation is radiation that escapes from the minimum-z boundary of the problem cylinder.
- c) Laterally escaping radiation is radiation that escapes from the curved lateral surface of the problem cylinder.

These definitions are maintained regardless of how the source is defined.

Appendix E: ACCEP Codes - Geometry

The ACCEP 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 code¹² 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^{8,9} 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.

E.1 Problem Geometry

With the ACCEP 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 four distinct steps:

- a) Define the location and orientation of each solid geometrical body required for specifying the input zones.
- b) Specify the input zones as combinations of these bodies.
- c) Specify the volumes of the subzones, if necessary.
- d) Specify the material in each input zone.

E.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:

- a) 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.

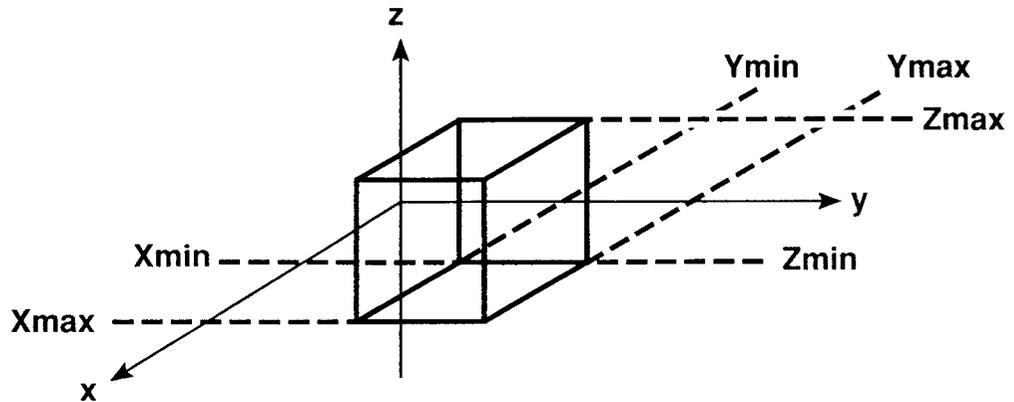


Figure E-1: Rectangular Parallelepiped (RPP)

- b) Sphere (SPH) -- Specify the components of the radius vector \underline{V} to the center of the sphere and the radius R of the sphere.

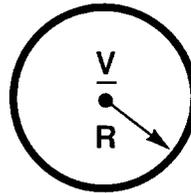


Figure E-2: Sphere (SPH)

- c) Right Circular Cylinder (RCC) -- Specify the components of a radius vector \underline{V} to the center of one base, the components of a vector \underline{H} from the center of that base to the center of the other base, and the radius R of the cylinder.

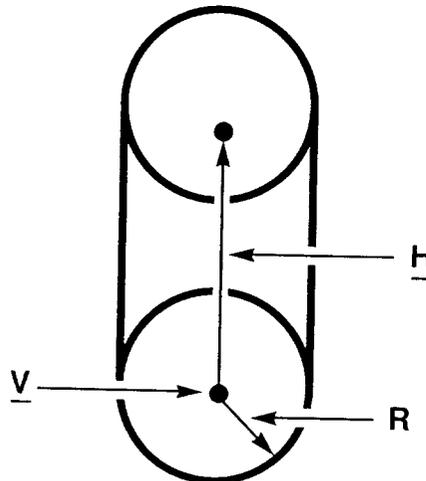


Figure E-3: Right Circular Cylinder (RCC)

- d) Right Elliptical Cylinder (REC) -- Specify the components of a radius vector \underline{V} to the center of one of the elliptical bases, the components of a vector \underline{H} from the center of that base to the center of the other base, and the components of two vectors \underline{R}_1 and \underline{R}_2 that define the major and minor axes, respectively, of the bases. This body has not yet been implemented.

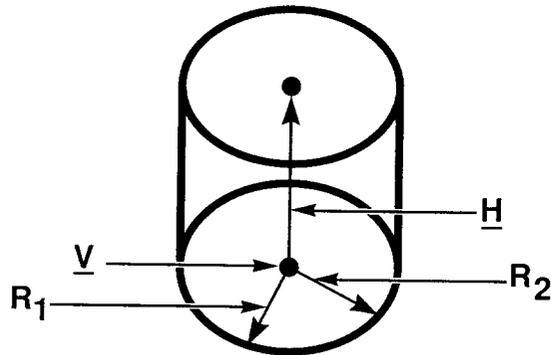


Figure E-4: Right Elliptical Cylinder (REC)

- e) Truncated Right Angle Cone (TRC) -- Specify the components of a radius vector \underline{V} to the center of one base, the components of a vector \underline{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.

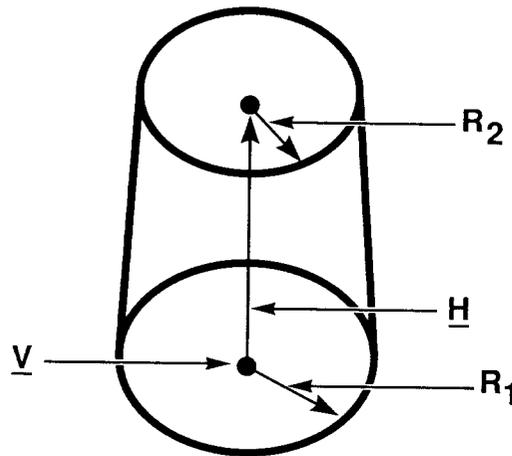


Figure E-5: Truncated Right Angular Cone (TRC)

- f) Ellipsoid (ELL) -- Specify the components of the radius vectors \underline{V}_1 and \underline{V}_2 to the foci of the prolate ellipsoid and the length of the major axis R .

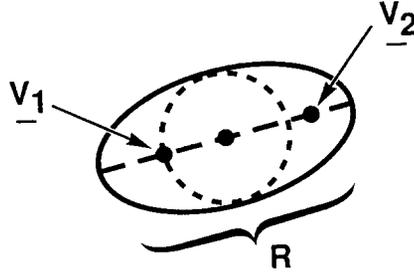


Figure E-6: Ellipsoid (ELL)

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

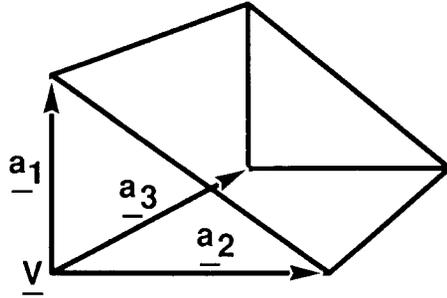


Figure E-7: Right Angle Wedge (WED)

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

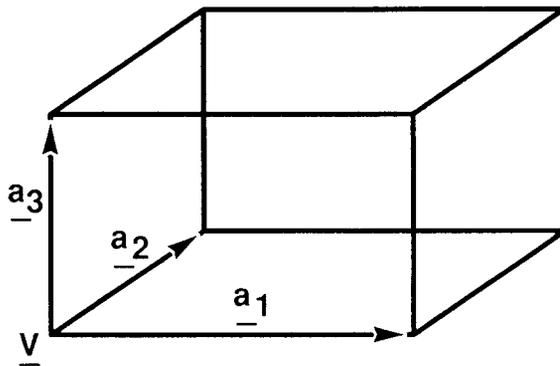


Figure E-8: Box (BOX)

- i) Arbitrary Polyhedron (ARB) -- Specify the components of k ($k = 6, 7, \text{ or } 8$) radius vectors, \underline{V}_1 through \underline{V}_k , to the corners of an

arbitrary nonreentrant 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.

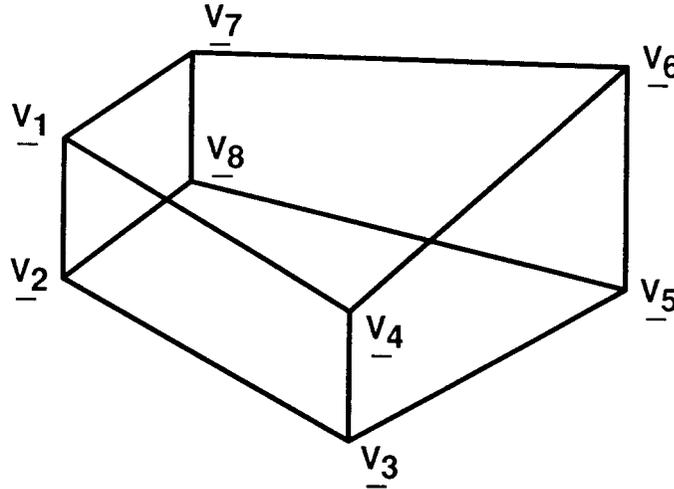


Figure E-9: Arbitrary Polyhedron (ARB)

E.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:

- a) An input zone may consist only of either a single homogeneous material or a void.
- b) Every point of the problem geometry must lie within one and only one input zone.
- c) 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 tallied 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. E-10.

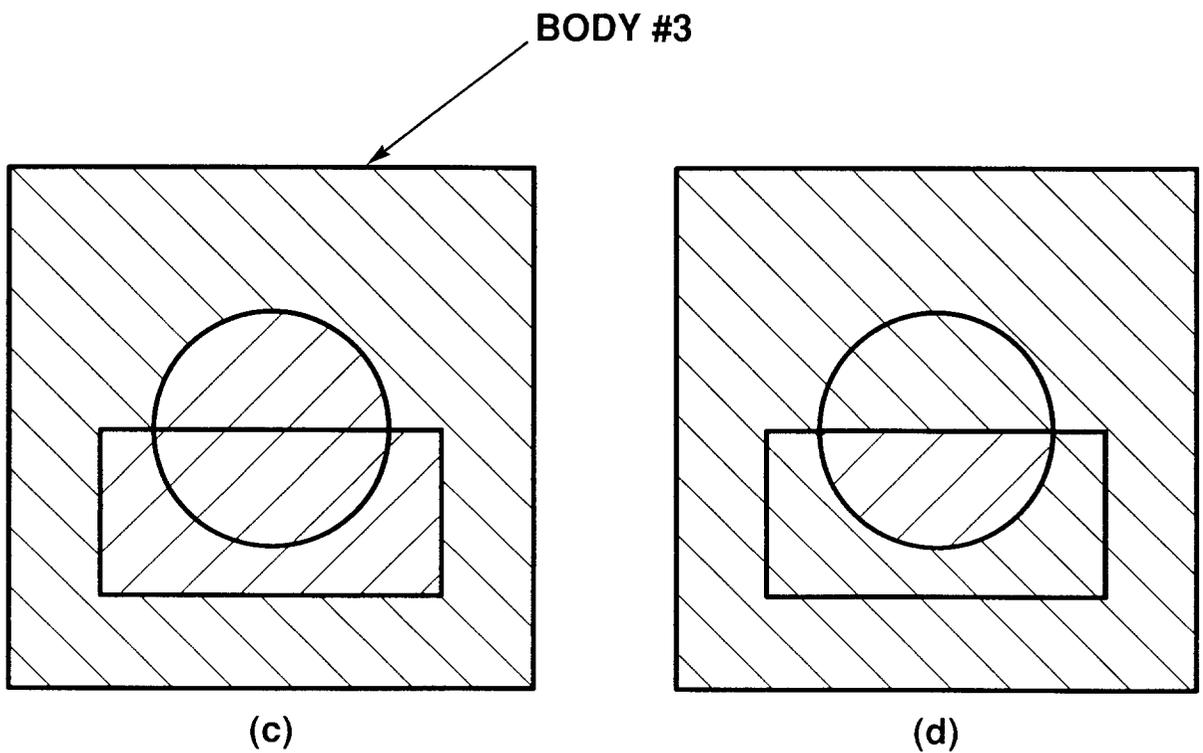
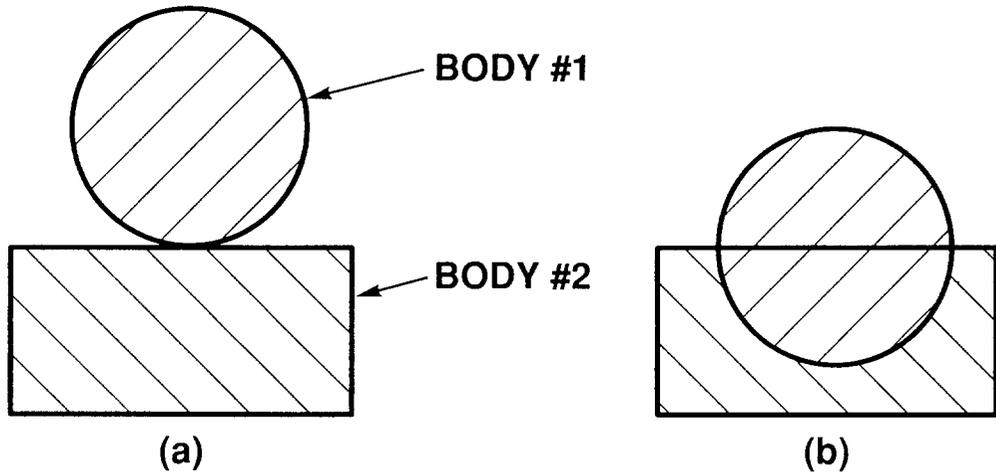
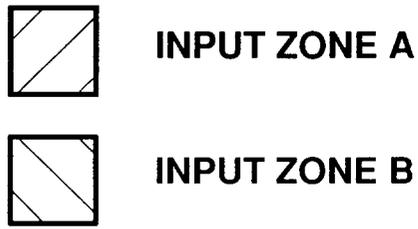


Figure E-10: Illustration of Various Methods of Combining Bodies for Specification of Input Zones.

In Fig. E-10a, 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

$$\begin{aligned} A &= +1 \\ \text{and } B &= +2 . \end{aligned}$$

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

In Fig. E-10b, the sphere is inserted into a hole that has been cut in the cylinder so that

$$\begin{aligned} A &= +1 \\ \text{and } B &= +2 -1 . \end{aligned}$$

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. E-10c, 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

$$\begin{aligned} A &= +1 \text{ OR } +2 \\ \text{and } B &= +3 -1 -2 . \end{aligned}$$

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. E-10d, 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

$$\begin{aligned} A &= +1 +2 \\ \text{and } B &= +3 -1 \text{ OR } +3 -2 . \end{aligned}$$

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

Note that:

- a) The OR operator refers to all following body numbers until the next OR operator is reached or a new input zone is initiated.
- b) 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. E-10a and E-10b 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.

E.1.3 Volume Specification -- A minor but important change in the specification of volumes has been initiated in Version 3.0. In all earlier versions, the volumes of the input zones were specified, whether set internally or supplied by the user. In Version 3.0, the volumes of the problem subzones are to be specified. Of course, if automatic subzoning has not been requested for any input zone, the input zones are the same as the problem subzones. Another minor point is that the volume of the escape zone is not specified. The key point here is that 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 one of three options via [parameter(1)] associated with the GEOMETRY keyword. The default value of 0 will cause the code to set all volumes to 1.0 cm³. A value of 1 allows the user to read in the volumes as described below. Finally, 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 correction run to insert the necessary logic at the proper place near the end of Subroutine JOGEN. The proper location is commented in the coding.

E.1.4 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 as described in Sec. 2.2.2. The method of inputting the material indices is described under the keyword GEOMETRY.

E.2 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 Secs. B.3, B.6 and B.8 of Appendix B.

E.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 Sec. E.1.1 and illustrated in Table E-1. 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 E-1 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 beginning with the parameter END signals that the description of all of the problem bodies is complete.

E.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. E.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. This parameter is followed by a string of parameters that specifies the input

Table E-1: Data Required to Describe Each Body Type

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

Note: The final line of the arbitrary polyhedron input contains a four-digit number for each of the 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. See Section E.1.1 for more details.

zone following the form of the right hand sides of the equations of Sec. E.1.2. For example, input lines describing the two input zones in Fig. E-10d 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 beginning with the parameter END signals that the description of all of the problem input zones is complete.

In Version 2.0 we began implementing the automatic subzoning features described in Appendix L into the ACCEPT codes. 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, or SPH. The automatic subzoning capability is invoked in the following way. All input zones to be subzoned are read in before those that are not to be subzoned. The first parameter on the line that begins the description of a zone to be subzoned must begin with the character string ZSUB rather than the single character Z for zones that are not to be subzoned. An input zone to be subzoned must consist wholly of a single body. In Version 3.0, that body type can only be RCC, RPP, BOX, or SPH. The line following the description of the input zone must contain three integers that define the number of equal-increment subzones into which the body is to be divided along the three coordinate directions. The following is a description of how the three integers are used for the four body types:

- 1) 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. E-3, respectively. The azimuthal angle is defined with respect to the vector ϵ_1 of the cartesian unit-vector triad, $(\epsilon_1, \epsilon_2, h)$, satisfying the right-hand rule, where h is a unit vector in the direction of H . In general, ϵ_1 is in the direction $j \times h$ so that when $h = k$, the triad degenerates to (i, j, k) . For the special case where h is paraxial with the y axis, the triad is $(i, -k, h)$ or (i, k, h) for $h = j$ or $h = -j$, respectively.
- 2) 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 $(X_{min}, Y_{min}, Z_{min})$ defined in Fig. E-1.
- 3) BOX -- The three integers correspond to subzoning along the three cartesian directions, a_1 , a_2 , and a_3 , respectively, as defined in Fig. E-8. Distances are measured from the point defined by the radius vector V .

- 4) 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.

E.2.3 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 geometry of the input zones. 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)] is not equal to 1, these input data are omitted.

E.2.4 Material Data -- For input of material data, return to the discussion under the GEOMETRY keyword.

Appendix F: P Codes

The SANDYL¹² 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.^{10,12}

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. 10, 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.

Appendix G: 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.^{13,37,38} Briefly stated, the rectilinear random-walk substeps of the field-free model^{1,2} are replaced by numerically-integrated segments of field trajectories in vacuo 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 algorithms³⁹ 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),¹⁴ 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. 38.

The current algorithm includes a major improvement over the method described in Ref. 14. 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.¹⁴ The algorithm now includes an extended RKF procedure¹⁵ 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.

Appendix H: 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 a few biasing options in ITS that are easily accessed via the input described in Sec. 2.3.2 and that have proven very useful in specific applications.

H.1 Trapped Electrons

In certain problems where only electrons that cross certain boundaries are important, the option activated by the TRAP-ELECTRONS keyword may be employed to reduce run time significantly. The parameter associated with this keyword is the global electron trapping energy. Internally, an array of zone-dependent electron trapping energies is obtained, each element of which is the greater of this global trapping energy or the zone-dependent cutoff energy for that particular zone (see Sec. H.6). The option becomes effective when an electron is trapped, that is, does not have enough energy to escape from a zone (or 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 zone (or 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. In Version 3.0 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 code-zone (regions separated by an OR operator in the input-zone definitions and input zones defined without the OR operator; see Sec. E.1.2) boundaries in the ACCEPT codes.

H.2 Scaling of Bremsstrahlung Production

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 keyword. The cross sections of the material chosen by the user are scaled so that an electron slowing down from the maximum source energy to the global electron cutoff energy in this material will, on the average, generate a number of bremsstrahlung photons equal to the first parameter associated with this keyword. The resulting scale factor is used to scale the bremsstrahlung cross sections for all other materials in the problem. The material selected as the basis for scaling (the second parameter associated with the SCALE-BREMS keyword) should be that material which one would expect to dominate the bremsstrahlung production. Simultaneous scaling of the cross section for electron impact ionization probability is also desirable (see Sec. H.3).

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 is employed to reduce the number of secondary electrons generated from the interaction of this artificially high bremsstrahlung 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 and charge deposition, and electron flux in regions of the problem where these profiles are determined by the transport and secondary electron production of the bremsstrahlung radiation. In such cases, the ELECTRON-RR secondary keyword should be used to ensure that a sufficiently large fraction of the secondary electrons are followed.

H.3 Scaling the Probability for Electron Impact Ionization

An option similar to that of the previous subsection 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 keyword. 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 keyword.

H.4 Forced Photon Collisions (TIGER and CYLTRAN Codes)

In the TIGER and CYLTRAN codes, an option is available for forcing a selected fraction of photons entering a given zone or leaving a collision site within that zone to interact in that zone. The option is activated via input data for the GEOMETRY keyword. 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 GEOMETRY keyword. 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).

H.5 Photon Path Length Stretching (ACCEPT Codes)

With the ACCEPT codes, the user has the ability to decrease (or increase) the photon interaction probability in a given input zone relative to its natural value by a different method from that used in the TIGER and CYLTRAN codes. The interaction probability is changed via the appropriate stretching (or shortening) parameter under the GEOMETRY keyword. The photon cross section for that input zone will be scaled by the reciprocal of that parameter. Although the parameter may be any real positive number between zero and one for shortening or between one and infinity for stretching, care must again be taken not to overbias.

H.6 Zone-Dependent Electron Cutoff Energies

The user has the option of varying the electron cutoff energy from zone to zone so long as the zone-dependent cutoffs are greater than or equal to the global electron cutoff energy. The option is activated via the appropriate parameter associated with the GEOMETRY keyword. When the energy of an 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 keyword 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 either of 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.

This option has proven useful in problems that involve the generation of bremsstrahlung in one zone(s) and deposition caused by that bremsstrahlung in another zone(s). 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 or she may not want to kill those electrons with Russian Roulette), or he or she may not want electron transport in those zones at all (bulk deposition).

H.7 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 (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). This was remedied in earlier versions of the TIGER and CYLTRAN codes by using instead 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.

In Version 3.0, the next-event estimator for photon leakage in the TIGER and CYLTRAN codes has been extended to the ACCEPT codes. This option is now 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 keyword, NEXT-EVENT-ESCAPE.

Appendix I: Plotting Capability

The user will find a plotting capability to be a useful tool in applications of the ACCEPT and CYLTRAN codes, and especially so when the MCODES option has been selected. Once a particular plot package has been implemented, the plots are easily obtained by using the *DEFINE,PLOTS instruction described in Sec. 2.3.1 and the PLOTS keyword described in Sec. 2.3.2.

I.1 CYLTRAN and ACCEPT Code Geometries

When calculating radiation transport in complex 3-D geometries, it is important to have one or more independent methods for verifying the accuracy of the geometry input data. Visual checking of input data can be very tedious and errors are likely to be overlooked. Since automatic internal volume calculations are not possible with combinatorial geometry, volume checks cannot be used to verify the accuracy of the geometry input for the ACCEPT codes as they can, for example, for the CYLTRAN codes. ITS includes the logic for obtaining plots of the geometry of the ACCEPT codes. The plots will quickly reveal any significant errors in the numerical input.

Briefly stated, these routines allow the user to obtain an arbitrary number of parallel projections of his body specification along arbitrary viewing directions. The final body, which usually encloses the rest of the geometry and defines the escape zone (see Appendix E), is not plotted. The user inputs the two spherical polar angles defining the direction from which the geometry is to be observed. The code then internally calculates a third rotation about k'' (the positive z direction after two rotations defined by the two input angles) such that

$$j''' \cdot (j \times k'') = 0$$

and

$$j''' \cdot j \geq 0.$$

Consequently, the original y axis will project onto the y''' axis with the same sense; that is, a vector pointing in the original y direction will be pointing in the y''' direction in the $x'''-y'''$ plots. The resulting projection operator is represented by a rotation matrix that is the product of three rotations. These rotations are quite similar to the Eulerian angles that are often used to describe the kinematics of rigid body motion.⁴⁰ Version 3.0 of ITS includes logic for plotting seven of the eight active primitive body types of the ACCEPT codes.

In the case of the CYLTRAN codes, one simply obtains a $\rho-z$ plot of all or part of the problem geometry.

I.2 Electron and Positron Trajectories (M Codes)

The M codes are most often applied to problems in which macroscopic field transport competes strongly with collisional transport. Indeed, electric fields are restricted to void regions. It is generally very helpful for the user to be able to view the effects of the nonstochastic macroscopic field transport. ITS includes the necessary logic for plotting a sample of electron and positron trajectories in regions of non-zero macroscopic fields. In the case of CYLTRANM, the trajectories are plotted on a ρ -z plot of all or part of the problem geometry. In the case of ACCEPTM, the trajectories appear on the final parallel projection as discussed in the previous subsection.

I.3 Implementation of Plot Packages

Version 3.0 of ITS is being distributed as a plot-capable but not a plot-ready program because no universal, machine-portable X-Y plotting package could be identified to include with this version. However, any installation that possesses an X-Y plot package can easily interface that package for use with the ACCEPT and CYLTRAN codes. We suggest using the smallest and most basic plot package in order to maximize the memory available for complex transport calculations. This is especially important when running on machines with limited fast memory such as the CDC CYBER 76. At Sandia we are using a small, local plot package called WEASEL.²⁶

The plotting functions in ITS have been partitioned into three categories: 1) device initiation, 2) axes drawing and curve plotting, and 3) device termination. These three functions correspond to three separate subroutines in ITS: 1) PLTBEG, 2) PLTDAT, and 3) PLTEND, respectively. The UPEML update program can be used to replace the existing calls to Sandia WEASEL routines in these subroutines with calls to the desired plot package. Then ITS can be compiled and linked with the plot library according to local conventions; the result is a plot-enabled version of ITS.

We conclude this section with a brief description of the function of each of these three subroutines. Subroutine PLTBEG simply calls the WEASEL routine WSTART. WSTART initializes the plot package and opens the appropriate output file. Subroutine PLTEND performs the opposite function. It calls WEND whose function is to terminate the plot package and close the output file. The remaining plot routine calls are contained in subroutines PLTDAT and PLOTA. Subroutine PLTDAT controls the drawing of the plot axes, plot title, and X-Y data. Furthermore, PLTDAT controls the creation of new plot frames through the calls to WFLUSH and WNEWPG. The meanings of the variables used in this routine are explained in the subroutine banner. Once again, one simply replaces a routine like WXAXIS with the corresponding routine that draws and labels an X axis, or replaces WDRAW with a call to a routine that plots the X-Y data on a predefined grid to generate a locally plot-enabled version of ITS. Additional logic was added to the end of Subroutine PLOTA for the purpose of drawing the orientation of the original X-Y-Z axes in the chosen rotated frame. The additional WEASEL-specific calls to WTEXT are needed to label these axes.

Appendix J: 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 10 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 10 by using the keyword BATCHES as described in Sec. 2.3.2.

A further benefit from this batch processing is a feature that prevents the user from exceeding his time limit. Before beginning a new batch, the remaining portion of the time requested for the job is obtained by a call to Subroutine TIMER and compared with an estimate of the time per batch. If this estimate is larger than the time remaining, results based on the number of completed batches -- including estimates of the statistical errors -- are printed out and the run is terminated.

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:

$$(S.E.)_N = \frac{100}{|\langle x_N \rangle|} \left| \frac{\langle x_N^2 \rangle - \langle x_N \rangle^2}{N - 1} \right|^{1/2},$$

where

$$\langle x_N \rangle = \frac{1}{N} \sum_{i=1}^N X_i,$$

and

$$\langle x_N^2 \rangle = \frac{1}{N} \sum_{i=1}^N X_i^2.$$

The X_i 's are the values of the quantity obtained from each batch, and N is the total number of completed batches (default 10).

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).

Appendix K: Notes on Job Control Language - (JCL)

The one site-specific aspect of running the ITS package at a given installation is the creation of appropriate Job Control Language (JCL) decks for a particular computer and operating system. We have had local experience running Version 3.0 of ITS under VAX/VMS, CRAY/UNICOS, CRAY/CTSS, workstations, and specialized mainframes. We have had local experience running Version 1.0 of ITS under VAX/VMS, CRAY/COS, and CDC/SCOPE. We have also had limited testing of Version 1.0 on an IBM/OS mainframe. However, we cannot hope to give JCL examples for all machines and all operating system since, for example, the JCL for CRAY/CTSS will be different from that for CRAY/COS. Therefore, we give generic examples for the JCL with special emphasis on which user files correspond to which Fortran unit numbers and Fortran logical file names in assignment statements. In section K.1 we give generic examples for creation runs, while in section K.2 we give examples for correction runs. Most modern operating systems should facilitate the easy translation of these generic steps into an efficient site-specific JCL. Our limited experience with the IBM/OS-type JCL, however, has indicated that it could present a significant stumbling block to the rapid implementation of the ITS package that we have envisioned for its machine-portable structure. Therefore, as a guideline for IBM users, in section K.3 we give some specific examples of IBM JCLs.

K.1 Generic Creation Runs

In this section we assume that the user has compiled and link/loaded the appropriate machine-dependent version of the UPEML utility and has saved it as an executable file. We also assume that the source files for XGEN and ITS have been copied from the magnetic tape into appropriately named permanent disk files.

Execute: Execute UPEML with command line input "I,N,L". (Note that the command line input is read from the default system input device, Fortran unit 5.)

Input: 1) The source file (XGEN or ITS) should be equivalenced to the file "srcdk" (alternatively, the user could avoid such equivalences by, e.g., replacing "I,N,L" of the command line with "I=XGEN,N=XGEN.PL,L=XGEN.LST"; see Sec. 3).

Work Files: None

Output: 1) A Fortran 77 direct access file that contains the new program library (NEWPL) created by this run. This file is written to the file "newpl" and should be appropriately renamed and permanently stored for subsequent use in all correction runs.

- 2) A listing of all the source card images that have been placed in the NEWPL along with the deckname and associated sequence numbers that have been assigned to each line. This listing is written to the file "listfl".

K.2 Generic Correction Runs

In this section we assume that the user has successfully created NEWPLs for both XGEN and ITS as described in the previous section and that these program libraries have been appropriately named and permanently stored for use in all correction runs. These program libraries will henceforth be referred to as old program libraries (OLDPLs). We also assume that the user has read the cross-section data file XDATA from magnetic tape and permanently stored it on disk. The generic-correction-run JCL that we will describe in this section will be appropriate for the combined runs of both XGEN and ITS in a single pass. As such, it will be compatible with the sample input found in Secs. B.1 through B.8 of Appendix B. The JCLs that we will describe would directly precede those input decks. We will clarify the relation of the input found in Appendix B to the execution steps in the following description.

STEP #1 - Make a compile file for XGEN.

Execute: Execute UPEML with command line input "F,P". (Note that the command line input is read from the default system input device, Fortran unit 5.)

Input:

- 1) The previously created and stored OLDPL for XGEN should be equivalenced to the file "oldpl" (or user could specify, e.g., "P=XGEN.PL").
- 2) Additional in-line input, as shown in Appendix B, is read from the default system input device, Fortran unit 5. (Note that if the I parameter is specified on the command line, this input will be read from the file "srcdk".) This information includes the UPEML command line, the machine and code definition information as described in Table III, and any additional user modifications to the code as described in Sec. 3.3. This information is contained between the first and second {eor}s in the examples of Appendix B.

Working Files: UPEML copies the OLDPL (the file "oldpl") information to a temporary working file, Fortran unit 15. This file is also direct access.

Output:

- 1) A compile file for XGEN is created as the file "cmpfil".
- 2) If an "L" (comma delimited) is added to the UPEML "command" line, a listing file is also created on the file "listfl". This file contains a summary of the correction run operations.

STEP #2 - Compile and link/load program XGEN.

Execute: Execute the system Fortran compiler and link loader to create an executable image of program XGEN.

Input: 1) The previously created Fortran compile file (written to the file "cmpfil") is used as input to the Fortran compiler.

Working Files: The Fortran compiler will create an object file that is then passed to the system link/loader.

Output: 1) An executable image of program XGEN is created for use in the next execution step.
2) Any Fortran or link/loader information requested by the user is written to the output file.

STEP #3 - Execute program XGEN and create electron/photon cross sections.

Execute: Execute the executable image of program XGEN to create electron and photon cross sections for use by the ITS Monte Carlo transport code.

Input: 1) The previously created image of program XGEN is executed.
2) The stored cross-section data file XDATA should be assigned to Fortran unit 9.
3) Additional in-line input, as shown in Appendix B, is read from the default system input device, Fortran unit 5. This is the problem-dependent input for XGEN and is contained between the second and third {eor}s in the examples of Appendix B.

Working Files: If the PRINT-ALL option is not requested as an option in the XGEN input cards then XGEN discards the detailed cross-section information written to Fortran unit 7.

Output: 1) An electron/photon cross-section file for input to ITS is written to Fortran unit 11.
2) A summary of the XGEN output is written to the default system output device, Fortran unit 6.

STEP #4 - Make a compile file for ITS.

Execute: Execute UPEML with command line input "F,P". (Note that the command line input is read from the default system input device, Fortran unit 5.)

Input: 1) The previously created and stored OLDPL for ITS should be equivalenced to the file "oldpl" (or user could specify, e.g., "P=ITS.PL").

- 2) Additional in-line input, as shown in Appendix B, is read from the default system input device, Fortran unit 5. (Note that if the I parameter is specified on the command line, this input will be read from the file "srcdk".) This information includes the UPEML command line, the machine and code definition information as described in Table V, and any additional user modifications to the code as described in Sec. 3.3. This information is contained between the third and fourth {eor}s in the examples in Appendix B.

Working Files: UPEML copies the OLDPL (the file "oldpl") information to a temporary working file, Fortran unit 15. This file is also direct access.

- Output:
- 1) A compile file for ITS is created as the file "cmpfil".
 - 2) If an "L" (comma delimited) is added to the UPEML command line, a listing file is also created as the file "listfl". This file contains a summary of the correction run operations.

STEP #5 - Compile and link/load program ITS.

Execute: Execute the system Fortran compiler and link loader to create an executable image of program ITS.

- Input:
- 1) The previously created Fortran compile file (written as the file "cmpfil") is used as input to the Fortran compiler.

Working Files: The Fortran compiler will create an object file that is then passed to the system link/loader.

- Output:
- 1) An executable image of program ITS is created for use in the next execution step.
 - 2) Any Fortran or link/loader information requested by the user will be written to the output file.

STEP #6 - Execute program ITS and perform the transport calculation.

Execute: Execute the executable image of program ITS to calculate the transport of electrons and photons through the geometry of interest.

- Input:
- 1) The previously created image of program ITS is executed.
 - 2) The previously created cross-section file should be assigned for input as Fortran unit 11.
 - 3) Additional in-line input, as shown in Appendix B, is read from the default system input device, Fortran unit 5. This is the problem-dependent input for ITS and follows the fourth {eor} in the examples in Appendix B.

- 4) In the case of a restart, the data contained in the dump file from a previous run is read from Fortran unit 14.

Working Files: If the PRINT-ALL option is not requested as an option in the ITS input cards, ITS discards the intermediate batch output to Fortran unit 12.

- Output:
- 1) The results of the transport calculation are written to the default system output device, Fortran unit 6.
 - 2) If the DUMP keyword is specified in the user input, a restart dump file will be written to Fortran unit 10.

K.3 Examples of IBM JCL

As discussed previously, we feel that information required by the IBM batch operating system is sufficiently detailed to warrant special consideration. We therefore present the following examples to guide the user in creating his own JCLs. We do not guarantee that these JCLs are completely site-portable; rather, they are simply guides. Note that for simplicity we have shown separate JCLs for each individual code step. We have not duplicated creation, correction, and compilation JCLs for XGEN since they are identical (except for file names) to those for ITS.

Example of creation run for ITS:

```
//ZWJHBI1 JOB (SB012,BF3,15),TEST.RUN,TIME=(5,30),CLASS=0
/*JOBPARM Q=F
/* DELETES OLD OUTPUT FILE IF IT EXISTS *
//FINDDSN EXEC SRCHDS,DSN='ITS.OPL'
//KILLFILE EXEC IEFBR14,COND=(4,LT,FINDDSN.SEARCH)
//DELETE DD DSN=ITS.OPL,DISP=(OLD,DELETE)
/* EXECUTES UPDATE EMULATOR - UPEML*
//RUNJOB EXEC GOSTEP,REGION=1024K,LIB='UPEML.LIB.LOAD',PROG=MAIN,
//      COND=EVEN
//FT15F001 DD DSN=ITS.OPL,DISP=(NEW,CATLG,DELETE),
//      UNIT=DISK,DCB=(RECFM=F,BLKSIZE=1024),VOL=SER=SACC09,
//      SPACE=(1024,5000)
//FT17F001 DD DSN=ITS.SRC(ITS),DISP=SHR
//DATA5 DD *
I,N,L
//NOTIFY EXEC NOTIFYTS
```

Example of Correction Run for ACCEPT code from ITS package:

```
//ZWJHBI2 JOB (SB012,BF3,15),TEST.RUN,TIME=(10,30),CLASS=0
/*JOBPARM Q=F
/* DELETES OLD OUTPUT FILE IF IT EXISTS *
//FINDDSN EXEC SRCHDS,DSN='ITS.COMP'
//KILLFILE EXEC IEFBR14,COND=(4,LT,FINDDSN.SEARCH)
//DELETE DD DSN=ITS.COMP,DISP=(OLD,DELETE)
/* EXECUTES UPDATE EMULATOR - UPEML*
```

```

//RUNJOB      EXEC GOSTEP,REGION=1024K,LIB='UPEML.LIB.LOAD',PROG=MAIN,
//            COND=EVEN
//FT13F001 DD DSN=ITS.OPL,DISP=SHR
//FT20F001 DD DSN=ITS.COMP,DISP=(NEW,CATLG,DELETE),
//            DCB=(RECFM=FB,LRECL=90,BLKSIZE=2700),UNIT=DISK,
//            SPACE=(TRK,(10,10)),VOL=SER=SACC04
//FT15F001 DD DSN=&&TEMP,UNIT=SYSDA,DISP=(NEW,PASS,DELETE),
//            DCB=(RECFM=F,BLKSIZE=1024),SPACE=(1024,5000)
//DATA5      DD *
I,F,P
*DEFINE IBM
*DEFINE ACCEPT
//NOTIFY EXEC NOTIFYTS

```

Note that the correction run JCL creates a compile file that has 90 character records to accommodate the deck and sequence number information that UPEML appends to each line. However, our experience has indicated that the IBM Fortran compiler insists on 80 character records. Therefore, we needed to truncate the 90 character ITS.COMP file to an 80 character file that we named ITS.FORT for the next job step.

```

Example of compilation and linking of the ACCEPT code of the ITS system:
//ZWJHBCMI JOB (SB012,BF3,5),TEST.RUN,TIME=(1,30),CLASS=0,
//            MSGCLASS=A,MSGLEVEL=(1,1),NOTIFY=ZWJHB
/*JOBPARM    Q=F,LINES=100
//COMPILE    EXEC FORTVCL,FVPOPT=2,FVPOLST='NOTF,NOLIST',
//            FVLNSPC='3200,(200,100)',REGION=4096K,
//            PARM.LKED='LET,MAP,XREF,NCAL'
//FORT.SYSIN DD DSN=ITS.FORT,DISP=SHR
//LKED.SYSLIB DD DSN=SYS1.VFORTLIB,DISP=SHR
//            DD DSN=SYS2.IMSL,DISP=SHR
//LKED.SYSLMOD DD DSN=ITS.LIB.LOAD(ACCEXE),DISP=SHR,LABEL=(,,OUT)
//NOTIFY EXEC NOTIFYTS

```

Example of running the cross-section generation program XGEN:

```

//ZWJHBX     JOB (SB012,BF3,15),TEST.RUN,TIME=(5,30),CLASS=0,
//            MSGCLASS=A,MSGLEVEL=(1,1),NOTIFY=ZWJHB
/*JOBPARM    LINES=100,Q=F
//FINDDSN    EXEC SRCHDS,DSN='XSECS.DATA'
//*          CREATE AND ALLOCATE SPACE FOR CROSS-SECTION FILE
//CREATE     EXEC IEFBR14,COND=(4,GT,FINDDSN.SEARCH)
//ALLOCATE DD DSN=XSECS.DATA,UNIT=DISK,DISP=(NEW,CATLG),
//            DCB=(RECFM=FB,LRECL=120,BLKSIZE=2400),
//            VOL=SER=SACC08,SPACE=(TRK,(5,5))
//XGENEXE    EXEC GOSTEP,REGION=2048K,LIB='XGEN.LIB.LOAD',
//            PROG=XGEN,COND=EVEN
//FT07F001 DD DUMMY                                DETAILED CROSS-SECTION INFO
//FT09F001 DD DSN=XDATA.DATA(XDATA),DISP=SHR,LABEL=(,,IN)

```

```

//FT11F001 DD DSN=XSECS.DATA,DISP=SHR,LABEL=(,,OUT)
//DATA5 DD *
MATERIAL TA
MATERIAL AL
TITLE
  1.0 MEV CROSS SECTIONS FOR TA AND AL
ENERGY 1.0
/*
//NOTIFY EXEC NOTIFYTS

```

Example of executing the ACCEPT member code of ITS:

```

//ZWJHBITS JOB (SB012,BF3,20), 'ACCEPT RUN', CLASS=0, TIME=(2,00),
//      MSGCLASS=A, MSGLEVEL=(1,1), NOTIFY=ZWJHB
/*JOBPARM RESTART=Y, Q=F, LINES=100
//ITSEXE EXEC GOSTEP, REGION=2048K, LIB='ITS.LIB.LOAD',
//      PROG=ACCEXE
//FT07F001 DD DUMMY
//FT06F001 DD SYSOUT=*, DCB=(RECFM=VBA, LRECL=137, BLKSIZE=137)
//FT11F001 DD DSN=XSECS.DATA, DISP=SHR, LABEL=(,,IN)
//FT12F001 DD DUMMY
//GO.DATA5 DD *
ECHO 1
TITLE
...1.0 MEV TA/AL TEST PROBLEM
***** GEOMETRY *****
GEOMETRY
  RCC      0.00      0.00      0.00      0.00      0.00      -0.007
           10.0
  RCC      0.00      0.00      0.00      0.00      0.00      0.10
           10.0
  RCC      0.00      0.00      0.10      0.00      0.00      0.05
           10.0
  SPH      0.00      0.00      0.00      12.0
END
  ZSUB1   +1
  ZSUB2   +3
  Z3      +2
  Z4      +4   -1   -2   -3
END
* MATERIAL CUTOFF STRETCHING
1
2
0
0

```

```

***** SOURCE *****
ELECTRONS
ENERGY 1.0
CUTOFFS 0.05 0.001
POSITION 0.0 0.0 -0.007
  RADIUS 2.50
* DEFAULT DIRECTION
DIRECTION 0.0 0.0
***** OUTPUT OPTIONS *****
ELECTRON-ESCAPE
  NBINE 2
  NBINT 4
PHOTON-ESCAPE
  NBINE 2
  NBINT 4
ELECTRON-FLUX 1 2
  NBINE 2
PHOTON-FLUX 1 2
  NBINE 2
***** OTHER OPTIONS *****
HISTORIES 1000
BATCHES 10
*BATCHES 20
* ... X-RAY PRODUCTION SCALING
SCALE-BREMS 500.
* PLOTS 3
* -15.0  15.0  -15.0  15.0  180.  90.
* -15.0  15.0  -15.0  15.0  0.0  0.0
* -15.0  15.0  -15.0  15.0  180.  30.
/*
//MESSAGE EXEC NOTIFYTS, ID=ZWJHB

```

Appendix L: Automatic Subzoning

Automatic subzoning refers to that feature of the ITS codes whereby the user may direct a particular member code to internally subdivide 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 have substantially extended this feature. We have implemented full automatic subzoning in the CYLTRAN codes, extending it to three dimensions with the addition of azimuthal subzoning. Moreover, the ACCEPT codes now feature the full three-dimensional subzoning of input zones consisting of a single body of type RCC, RPP, BOX, or SPH.

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. In Version 3.0 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.

Currently, the presence or absence of subzone-based testing for electron trapping (see Appendix H.1) as in Item (3) depends on the particular code being used. In the TIGER codes, it is fully subzone based. In the CYLTRAN codes, it is subzone based for axial and radial subzones, but not for azimuthal subzones. In the ACCEPT codes, it is based only on the input zones. The potential problem with testing only on input zone boundaries is that the energy and charge of an electron found to be so trapped is deposited on the spot, even though its residual range may still be substantially greater than the dimensions of the subzones within that input zone. For now we recommend one of two approaches for such cases. Users may simply not use the TRAP-ELECTRONS option and choose a zone-dependent electron cutoff energy (see GEOMETRY keyword) for the input zone to be subzoned such that the residual range at this cutoff is of the order of or less than the minimum dimension of a subzone. Or they may use the TRAP-ELECTRONS option and explicitly define as input zones what would otherwise have been subzones. We hope to make all testing for trapped electrons subzoned based in future versions of ITS, but this must be done without adding an excessive amount of overhead from the required analytic geometry calculations.

The remaining issues 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 use in Version 3.0 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 the UPEML processor 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 derives 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 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 the UPEML processor, this logic may easily be extended to the calculation of electron flux in the TIGER codes.

Appendix M: Cross-Section Improvements in Version 3.0

M.1 Bremsstrahlung Cross Sections

In Version 3.0, an improved, comprehensive database of electron-bremsstrahlung production cross sections, differential in emitted photon energy, has been incorporated into the cross-section generator. The development of these data, which covers electrons with kinetic energies from 1 keV to 10 GeV incident on neutral atoms with atomic numbers $Z = 1$ to 100, is discussed in rather complete detail by Seltzer and Berger^{41,42} and by Seltzer.⁴³ The following is a brief description of the data base.

The cross sections were prepared by combining various theoretical results that go beyond the Bethe-Heitler, Born-approximation results largely relied on in the cross-section package of Version 2.1. For bremsstrahlung in the coulomb field of the atomic nucleus, the new cross sections are based on (a) the results from numerical phase-shift calculations for a screened coulomb potential by Pratt et al.⁴⁴ at energies below 2 MeV and (b) an analytical high-energy theory derived with coulomb wave functions of Davies et al.⁴⁵ and Olsen⁴⁶ at energies above 50 MeV, supplemented by the Elwert coulomb correction factor⁴⁷ and the theory of the high-frequency limit given by Jabbur and Pratt.⁴⁸ In the high-energy region, the effect of screening was included by the use of Hartree-Fock atomic form factors from Hubbell et al.^{49,50}

A numerical interpolation scheme, applied to suitably scaled cross sections, was used to bridge the gap between the low-energy and high-energy theoretical results and thus to obtain improved cross sections in the intermediate-energy region from 2 to 50 MeV. The accuracy of the interpolation was checked by comparing interpolated results with those from the exploratory partial-wave numerical calculations of Tseng and Pratt⁵¹ for 5- and 1-MeV electrons in Al and U; the comparisons, shown in Seltzer and Berger⁴¹ and Seltzer,⁴³ indicate very good agreement.

Bremsstrahlung in the field of the atomic electrons was calculated according to the theory of Haug,⁵² combined with (a) screening corrections evaluated on the basis of the high-energy theory of Wheeler and Lamb⁵³ in conjunction with the use of incoherent-scattering functions from Hubbell et al.,⁴⁹ and (b) a small coulomb correction from Maxon and Corman.⁵⁴

The data base accessed by the generator consists of the total (electron-nucleus plus electron-electron) cross sections for 57 values of the electron kinetic energy T extending from 1 keV to 10 GeV, for 30 values of the ratio of photon-to-electron energy k/T extending from 0 to 1, and for all atomic numbers Z from 1 to 100. The generator uses cubic-spline interpolation among the data to evaluate the cross section

for desired T and k/T values; the output probability distributions are now evaluated for a set of k/T values that provide for finer coverage than in previous versions and that extend down to lower photon energies.

Also added to the data base were the scaled, integrated energy-loss cross sections Φ_{rad} , which were prepared by integrating over the differential bremsstrahlung cross sections for each Z and for each of the 57 electron kinetic energies T. These quantities are used in the generator to evaluate the radiative stopping power. Comparisons of the new Φ_{rad} data to those obtained from the empirically-corrected Bethe-Heitler cross-section package (Berger and Seltzer⁵⁵) used in Version 2.1 are given by Seltzer⁴³ and show differences of 10-15% above about 1 MeV, increasing to much larger factors at low energies. The new results are essentially consistent, within less than 1% in most cases, with those used by Berger and Seltzer⁵⁶ in the preparation of ICRU Report 37.⁵⁷ The small differences are due largely to the omission in the ICRU work of bremsstrahlung produced by the recoiling electron in electron-electron interactions.

M.2 Electron and Positron Stopping Powers

The algorithms used by Berger and Seltzer⁵⁶ in ICRU Report 37⁵⁷ for the calculation of the collisional stopping power of electrons and positrons has been incorporated into the generator. See those references for a detailed description of the methods and data used in the development of these procedures; the following gives only brief highlights of the improvements.

The collisional stopping power is still calculated on the basis of Bethe stopping-power theory. In addition to simple atomic constants and kinematic terms, the stopping-power formula contains two non-trivial quantities: (1) the mean excitation energy I, the key material-dependent parameter in Bethe's theory, and (2) the density-effect correction, which accounts for the reduction of the stopping power due to the polarization of the medium. Improved knowledge of both of these quantities is used in the current methods.

A set of mean excitation energies for elemental substances with atomic numbers Z from 1 to 100 was prepared from a critical analysis of available experimental and theoretical information on oscillator-strength distributions, dielectric response functions, and charged particle stopping powers. From available data for gas, liquid, and solid compounds, and using a built-in data base of atomic weight from Holden,⁵⁸ an algorithm was developed that goes beyond simple Bragg additivity and takes into account, at least approximately, phase and chemical binding effects.

The density-effect correction is calculated directly, rather than relying on an analytical approximation of the results, according to the dispersion model of Sternheimer,⁵⁹ using atomic binding energies from Carlson.³⁰ Sternheimer's dielectric model, constructed to insure consistency with the input I-value, depends on the plasma frequency

obtained from the input value of the density of the medium. In our implementation we have included refinements that treat conductors and insulators differently. For conductors and semiconductors, the binding energy of conduction electrons, whose number is assumed to be the number of electrons occupying the most weakly bound shell of an isolated atom of each constituent element that would in its pure state also be considered a conductor or semiconductor, is set equal to zero with the consequence that a small density-effect correction extends down to arbitrarily low velocities. For insulators, the density-effect correction vanishes below a cutoff velocity.

Because of the updated I-values and the improved treatment of the density-effect correction, values of the collision stopping power can differ by as much as a few percent from those obtained from Version 2.1.

M.3 Energy-Loss Straggling

The collisional energy loss of an electron for a given pathlength is sampled in ITS from the Landau distribution.⁶⁰ By making the largely-justified assumption that the maximum energy loss can extend to infinity, Landau obtained the energy-loss distribution in terms of a universal function ϕ of a single scaled variable λ . Because of this assumption, there formally exists no finite mean for the Landau distribution. However, a mean is implied in practice, as the sampling is done over some range of λ . Versions 1.0 and 2.0 of ITS made sole use of Börsch-Supan's tabulation of the Landau distribution,⁶¹ which extends from $\lambda = -4$ to only $\lambda = 100$. Such a cutoff, coupled with a kinematic cutoff used in ITS, results in a correct mean energy loss only in the vicinity of 1.0 MeV.

This problem was recently corrected by simply imposing an upper limit of λ_{cut} on the sampling procedure that provides for the correct mean energy loss (see Seltzer⁶² for a more complete discussion, including the relationship between λ and energy loss Δ). To implement this procedure, the numerical tabulation of the Landau distribution was extended to a λ -value of 5×10^4 by using the asymptotic formula given by Börsch-Supan. This extended distribution was converted into 5001 equi-probable Landau λ 's for rapid sampling, and the new algorithm was implemented in Version 2.1 of ITS.

However, a systematic underestimation of the mean energy loss was found in the cases of $\lambda_{\text{cut}} \geq 1000$. This was traced to the poor representation of the large- λ tail provided by the 5001 equi-probable λ 's (only 5 λ 's were larger than 1000). This then led to the development of the Version-3.0 algorithm in which 5001 equi-probable λ 's are used to describe the distribution only from $\lambda = -4$ to 100, accounting for 0.9896 of the total distribution, combined with direct sampling from the large- λ tail for the remainder.

Blunck and Leisegang⁶³ included the second moment in the expansion at low energies of the energy-loss cross section used in solving Landau's equation. In Version 1.0 of ITS the collisional energy loss for each major macroscopic step is sampled from the Landau/Blunck-Leisegang distribution. However, for short pathlengths when the Blunck-Leisegang broadening becomes too large (see Seltzer⁶²), the distribution becomes inaccurate. This occurs in ITS at lower energies in high-Z targets. In Version 3.0, an empirical correction due to Seltzer⁶⁴, loosely based on the work of Chechin and Ermilova⁶⁵, has been implemented that results in significant improvement of code predictions in this parameter range.

M.4 Photon Cross Sections

The latest NIST database of photon interaction cross sections, covering energies from 1 keV to 100 GeV and all elements $Z = 1$ to 100, has been incorporated into Version 3.0. This data base was constructed through the combination of incoherent scattering cross sections from Hubbell et al.,⁴⁹ coherent scattering cross sections from Hubbell and Øverbø,⁵⁰ photoelectric absorption cross sections from Scofield,⁶⁶ and pair-production cross sections from Hubbell et al.²⁹ In a departure from recent tabulations of Hubbell,^{67,68} the photoelectric absorption cross sections were not renormalized using the relativistic Hartree-Fock correction factors supplied by Scofield. Recent reviews of Saloman and Hubbell^{69,70} indicate that agreement with experiment is better when the renormalization is not done.

For compounds and mixtures the interpolation-summation algorithm from the work for Berger and Hubbell⁷¹ is used in the generator to combine the cross sections of the constituent elements for a comprehensive energy list including all absorption edges. The cross sections for the different processes are converted to the appropriate relative probabilities needed by the Monte Carlo code. In contrast to prior versions, the total attenuation coefficients prepared for input to Version 3.0 include coherent scattering, as this process is now included in the photon Monte Carlo simulation.

M.5 Incoherent Photon Scattering

Although the total probability for incoherent scattering was taken from cross sections that included binding corrections, prior to Version 3.0 the relative energy-angle distribution of scattered photons was sampled solely from the Klein-Nishina cross section, which assumes the target electron to be initially free and stationary. A new algorithm was developed for the sampling of the scattered photons in Version 3.0 that includes binding effects through the use of the same incoherent-scattering functions used by Hubbell in the preparation of the integrated-over-angle incoherent cross section. The criteria adopted in this development were that the methods be both formally exact and efficient. The following is a brief discussion of these methods.

The integrated incoherent scattering cross section is from Hubbell et al.⁴⁹ and is obtained by integrating over a differential cross section that is the product of the Klein-Nishina cross section and the incoherent scattering function. At high energies the angle of the scattered photon is sampled from the Klein-Nishina probability density function and accepted or rejected based on the incoherent scattering function. At low photon energies, however, this method becomes quite inefficient, because momentum transfers are restricted to small values where the incoherent scattering function is small. Consequently, at low energies the procedure is reversed, and the scattering angle is sampled according to a probability density function based on the incoherent scattering function and accepted with a probability given by the Klein-Nishina function. Performance tests using both methods were done as a function of atomic number and photon energy. From the results a switching energy, $E_s(\text{keV})=3.0|Z$, was found to approximately predict the incident photon energy at which the reversal in methods results in the fastest algorithm. It was found that for sampling from the Klein-Nishina probability density function, the use of Kahn's algorithm⁷² below about 5 MeV and Koblinger's method⁷³ above 5 MeV resulted in the fastest performance.

M.6 Photon Coherent Scattering

Angular deflections associated with coherent scattering were neglected in ITS prior to Version 3.0, using the argument that at energies where the contribution of coherent scattering becomes largest (of the order of only 10% of all photon interactions) the photon deflection angles tend to be rather small. To include coherent scattering, a sampling algorithm was developed for Version 3.0 along lines similar to those for incoherent scattering.

The integrated coherent scattering cross section is from Hubbell et al.⁴⁹ and is obtained by integrating over a differential cross section that is the product of the Thomson cross section and the atomic form factor. At low energies the angle of the scattered photon is sampled from the probability density function obtained from the Thomson cross section and accepted or rejected based on the atomic form factor. At high photon energies, however, this method becomes quite inefficient. Consequently, at high energies the procedure is reversed, and the scattering angle is sampled according to a probability density function based on the atomic form factor and accepted with a probability given by the Thomson cross section. The speed of the algorithm was found to be optimized by switching between the two methods at approximately $E_s(\text{keV})=2.0 Z^{0.2}$.

Appendix N: 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 believe we have done this for the machines we support (Cray, CDC, VAX and IBM main-frames). In this appendix, we share the issues of concern we have encountered over the years in order to assist the user in porting ITS to unsupported platforms (e.g., workstations).

N.1 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. While such values are not obtained from most generators, the RNG on the VAX, for example, can return the value of zero; in fact, it is capable of returning two consecutive zeroes. Logic was added to the VAX coding of the Monte Carlo program library to resample when a zero random number is generated.

N.2 Access to the Seed

The seed of a RNG is that variable which is used to determine the next random number in the sequence. 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, the starting random-number seed for the current source particle (variable IRSAV in COMMON /STTS/) is printed within the terminating subroutine, ABORTX.

If there is a problem accessing the seed (we have been told that this is the case with the SUN-4 workstations, for example), the user can delete references to RANINT and RANSAV from SUBROUTINE HIST at the penalty of no longer having the above-mentioned capabilities. One symptom of incorrect implementation 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.

N.3 Reproducibility

A RNG is designed to produce the same sequence of random numbers for the same starting seed. A problem may occur for 32-bit machines running in double precision while using a single precision RNG. Care must be taken to insure zero fill, instead of using extraneous, nonreproducible bits from memory, when performing the conversion from the single precision sample (output from the RNG) to the double precision number to be used in the computations.

This problem was encountered with the VAX RNG in Version 1.0 of ITS. While it has no effect on the accuracy of the results, for debugging purposes it is useful to be able to precisely reproduce a single history when needed. The extra logic in the current version of function RAN, for the VAX only, involving the EQUIVALENCE of a double precision variable with a two-element single precision array, was our solution. Note that this logic takes advantage of the way the VAX represents single and double precision numbers; it will not be the same for all machines. For example, this scheme will not work for the IBM RISC System/6000 series of UNIX workstations.

N.4 Cycle Length

Computer generated random numbers have a finite cycle length (the number of random numbers generated before the cycle repeats itself). One should not run so many source particles in a given run so as to exceed the RNG cycle length. The statistical uncertainties from such a run can be misleadingly low. A single source particle will likely use many random numbers. The cycle length on the VAX is about 2^{32} ; however, this will vary from one vendor to the next.

N.5 Portable Random Number Generator

We have successfully implemented a version of the machine-independent RNG used in the MCNP code,⁷⁴ which has a cycle length of 2^{46} , never generates zeroes, allows easy access to its seed, and is already double precision. This RNG also relieves the user of concerns about the integrity and implementation of the intrinsic RNG on his system. However, there is an increase in run time (about 33% on the VAX). This portable RNG may be invoked in place of the default RNG by using the *DEFINE,RNG1 input directive to the UPEML processor.

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, June 1968.
2. M. J. Berger, "Monte Carlo Calculation of the Penetration and Diffusion of Fast Charged Particles," Methods in Computational Physics, Vol. 1, (Academic, New York 1963).
3. J. A. Halbleib and W. H. Vandevender, "EZTRAN--A User-Oriented Version of the ETRAN-15 Electron-Photon Monte Carlo Technique," SC-RR-71-0598, Sandia National Laboratories, September 1971.
4. J. A. Halbleib and W. H. Vandevender, "EZTRAN 2: A User-Oriented Version of the ETRAN-18B Electron-Photon Monte Carlo Technique," SLA-73-0834, Sandia National Laboratories, November 1973.
5. J. A. Halbleib and W. H. Vandevender, Nucl. Sci. Eng. 57, 94 (1975).
6. J. A. Halbleib and W. H. Vandevender, Nucl. Sci. Eng. 61, 288 (1976).
7. J. A. Halbleib, Nucl. Sci. Eng. 75, 200 (1980).
8. 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," MAGI-6701, Mathematical Applications Group, Inc., August 1967.
9. E. A. Straker, W. H. Scott, Jr. and N. R. Byrn, "The MORSE Code with Combinatorial Geometry," SAI-72-511-LJ (DNA 2860T), Science Applications, Inc., May 1972.
10. J. A. Halbleib and J. E. Morel, Nucl. Sci. Eng. 70, 219 (1979).
11. J. A. Halbleib and J. E. Morel, Sandia National Laboratories, unpublished.
12. H. M. Colbert, "SANDYL: A Computer Code for Calculating Combined Photon-Electron Transport in Complex Systems," SLL-74-0012, Sandia National Laboratories, May 1974.
13. J. A. Halbleib, Sr., and W. H. Vandevender, J. Appl. Phys. 48, 2312 (1977)
14. L. F. Shampine, H. A. Watts and S. Davenport, SIAM Rev. 18, 376 (1976).

15. K. L. Hiebert and L. F. Shampine, "Implicitly Defined Output Points for Solutions of ODEs," SAND80-0180, Sandia National Laboratories, February 1980.
16. J. A. Halbleib, Sandia National Laboratories, unpublished.
17. J. A. Halbleib, Nucl. Sci. Eng. 66, 269 (1978).
18. J. A. Halbleib, Sandia National Laboratories, unpublished.
19. T. A. Mehlhorn and J. A. Halbleib, "Monte Carlo Benchmark Calculations of Energy Deposition by Electron/Photon Showers up to 1 GeV," Proceedings of a Topical Meeting on Advances in Reactor Computations, ISBN 0-89448-111-8, March 1983, p. 608.
20. J. A. Halbleib and T. A. Mehlhorn, "ITS: The Integrated TIGER Series of Coupled Electron/Photon Monte Carlo Transport Codes," Sandia National Laboratories Report No. SAND84-0573, November 1984.
21. J. A. Halbleib and T. A. Mehlhorn, "ITS: The Integrated TIGER Series of Coupled Electron/Photon Monte Carlo Transport Codes," Nucl. Sci. Eng., Vol. 92, No. 2, p. 338, February 1986.
22. J. A. Halbleib, Monte Carlo Transport of Electrons and Photons (Edited by T. M. Jenkins, W. R. Nelson, and A. Rindi), New York: Plenum Press, 1988, pp. 249-284.
23. ITS Code Package CCC-467, Radiation Shielding Information Center, Computer Code Collection, Oak Ridge National Laboratory, January 1985.
24. "UPDATE, VERSION 1, Reference Manual," Revision 11/23/81, Report No. 60449900, Control Data Corporation.
25. T. A. Mehlhorn and M. F. Young, "UPEML Version 3.0: A Machine-Portable CDC UPDATE Emulator," SAND92-0073, Sandia National Laboratories, March 1992.
26. P. A. Watterberg, "WEASEL Reference Manual," Internal Report, Sandia National Laboratories.
27. F. Biggs and R. Lighthill, "Analytical Approximations for X-Ray Cross Sections II," SC-RR-71 0507, Sandia National Laboratories, December 1971.
28. F. Biggs and R. Lighthill, "Analytical Approximations for Total Pair-Production Cross Sections," SC-RR-68-619, Sandia National Laboratories, September 1968.
29. J. H. Hubbell, H. A. Gimm and I. Øverbø, J. Phys. Chem. Ref. Data 9, 1023 (1980).

30. T. A. Carlson, Photoelectron and Auger Spectroscopy (Plenum Press, New York, 1975).
31. L. G. Haggmark, C. J. MacCallum and M. E. Riley, Trans. Am. Nucl. Soc. 19, 471 (1974).
32. R. M. Sternheimer and R. F. Peierls, Phys. Rev. B3, 3681 (1971).
33. H. Kolbensvedt, J. Appl. Phys. 38, 4785 (1967).
34. M. Gryzinski, Phys. Rev. 138, A305, A322, A336 (1965); Phys. Lett. 38A, 35 (1972).
35. D. H. Rester and W. E. Dance, "Electron Scattering and Bremsstrahlung Cross Sections," CR-759, National Aeronautics and Space Administration (April 1967).
36. H. Aiginger, Z. Fur Physik 197, 8 (1966).
37. J. A. Halbleib, J. Appl. Phys. 45, 4103 (1974).
38. J. A. Halbleib and W. H. Vandevender, IEEE Trans. Nucl. Sci. NS-22, 2356 (1975).
39. J. A. Halbleib, Bull. Am. Phys. Soc. 26, 1062 (1981).
40. H. Goldstein, Classical Mechanics (Addison-Wesley, Reading, Massachusetts, 1959), p. 107.
41. S. M. Seltzer and M. J. Berger, Nucl. Instr. Meth B12, 95 (1985).
42. S. M. Seltzer and M. J. Berger, Atom. Data and Nucl. Data Tables 35, 345 (1986).
43. S. M. Seltzer, in Monte Carlo Transport of Electrons and Photons (Plenum Press, NY), p. 81 (1988).
44. R. H. Pratt, H. K. Tseng, C. M. Lee, L. Kissel, C. MacCallum, and M. Riley, Atom. Data and Nucl. Data Tables 20, 175 (1977); errata in 26, 477 (1981).
45. H. Davies, H. A. Bethe, and L. C. Maximon, Phys. Rev. 93, 788 (1954).
46. H. Olsen, Phys. Rev. 99, 1335 (1955).
47. G. Elwert, Ann. Physik 34, 178 (1939).
48. R. J. Jabbur and R. H. Pratt, Phys. Rev. 133, B1090 (1965).

49. J. H. Hubbell, W. J. Veigele, E. A. Briggs, R. T. Brown, D. T. Cromer, and R. J. Howerton, *J. Phys. Chem. Ref. Data* 4, 471 (1975); errata in 6, 615 (1977).
50. J. H. Hubbell and I. Øverbø, *J. Phys. Chem. Ref. Data* 8, 69 (1979).
51. H. K. Tseng and R. H. Pratt, *Phys. Rev.* A19, 1525 (1979).
52. E. Haug, *Z. Naturforsch.* 30a, 1099 (1975).
53. J. A. Wheeler and W. E. Lamb, *Phys. Rev.* 55, 858 (1939); errata in 101, (1956).
54. S. M. Maxon and E. G. Corman, *Phys. Rev.* 163, 156 (1967).
55. M. J. Berger and S. M. Seltzer, *Phys. Rev.* C2, 621 (1970).
56. M. J. Berger and S. M. Seltzer, National Bureau of Standards Report NBSIR 82-2550-A, (1983).
57. ICRU (International Commission on Radiation Units and Measurements) Report 37 (1984).
58. N. E. Holden, *Pure Appl. Chem.* 51, 405 (1979).
59. R. M. Sternheimer, *Phys. Rev.* 88, 851 (1952).
60. L. Landau, *J. Phys. (USSR)* 8, 201 (1944).
61. W. Börsch-Supan, *J. Res. National Bureau of Standards* 65B, 245 (1961).
62. S. M. Seltzer, in Monte Carlo Transport of Electrons and Photons (Plenum Press, NY), p. 153 (1988).
63. O. Blunck and S. Leisegang, *Z. Physik* 128, 500 (1950).
64. S. M. Seltzer, *Appl. Radiat. Isot.* 42, No. 10, pp. 917-941 (1991).
65. V. A. Chechin and V. C. Ermilova, *Nucl. Instr. Meth.* 136, 551 (1976).
66. J. H. Scofield, Lawrence Livermore National Laboratory Report UCRL-51326 (1973).
67. J. H. Hubbell, *Radiat. Res.* 70, 58 (1977).
68. J. H. Hubbell, *Int. J. Appl. Radiation & Isotopes* 33, 1269 (1982).
69. E. B. Saloman and J. H. Hubbell, National Bureau of Standards Report NBSIR 86-3431 (1986).

70. E. B. Saloman and J. H. Hubbell, Nucl. Instr. Meth. A255, 38 (1987).
71. M. J. Berger and J. H. Hubbell, Nation Bureau of Standards Report NBSIR 87-3597, (1987).
72. H. Kahn, Rand Corp. Research Memoradum RM-1237-AEC, (1956).
73. L. Koblinger, Nucl. Sci. Eng. 56, 218 (1975).
74. J. S. Hendricks, Nucl. Sci. Eng. 109, 86-91 (1991).

DISTRIBUTION:

1163 R. C. Hughes
1231 J. Maenchen
 Attn: staff
1231 T. W. L. Sanford
1238 D. L. Johnson
1240 K. R. Prestwich
1241 C. L. Olson
 Attn: staff
1241 J. W. Poukey
1243 E. L. Neau
1248 M. T. Buttram
1272 T. A. Mehlhorn
 Attn: staff
1272 L. P. Mix
1265 J. P. Quintenz
 Attn: staff
1265 M. A. Sweeney
1277 R. J. Leeper
 Attn: staff
1277 G. T. Baldwin
1277 D. L. Fehl
1270 J. K. Rice
1271 M. K. Matzen
 Attn: staff
1273 D. H. McDaniel
 Attn: staff
1275 R. A. Gerber
 Attn: staff
1290 T. H. Martin
1332 P. S. Winokur
1421 S. J. Plimpton
1820 J. Q. Searcy
1943 W. H. Vandevender
2520 N. J. Magnani
2525 R. J. Walko
3400 J. D. Martin
3437 R. G. Baca
5020 C. W. Childers
5023 C. A. Searls
 Attn: staff
5032 H. M. Sanger
5032 D. R. Waymire
5100 J. L. Wirth
5501 T. N. Simmons
6310 T. Blejwas
6320 R. E. Luna
6340 W. D. Weart
6410 D. A. Dahlgren
6420 W. B. Gauster
6428 R. T. McGrath

6450 T. R. Schmidt
6451 J. W. Bryson
6452 T. F. Luera
 Attn: staff
6453 D. Coats
 Attn: staff
6454 G. L. Cano
6460 J. V. Walker
6461 P. J. McDaniel
6462 P. S. Pickard
6462 D. A. McArthur
6465 J. H. Lee
 Attn: staff
6465 R. E. Pepping
6470 D. J. McCloskey, Actg.
6473 B. L. Spletzer
6620 D. L. Berry
7713 G. E. Tucker
 Attn: staff
7714 H. N. Jow
 Attn: staff
7722 G. E. Chavez
8341 A. J. Antolak
8347 K. L. Wilson
8433 M. H. Rogers
8541 A. M. F. Lau
9000 G. Yonas
9100 R. G. Clem
9210 H. M. Dumas
9220 G. H. Mauth
9230 L. S. Walker
9240 P. A. Stokes
9241 T. P. Wright
9241 S. A. Dupree
9300 J. E. Powell
9302 L. M. Choate
9310 J. D. Plimpton
9312 K. M. Glibert
 Attn: staff
9312 G. S. Mills
9312 T. J. Tanaka
9313 S. R. Dolce
9340 W. Beezhold
9341 J. R. Lee
 Attn: staff
9341 W. P. Ballard
9341 D. E. Beutler
9341 J. A. Halbleib (35)
9341 R. P. Kensek (35)
9341 D. Knott
9341 L. J. Lorence

9341 P. S. Raglin
9341 G. D. Valdez (20)
9343 V. Harper-Slaboszewicz
9350 J. H. Renken
9351 E. F. Hartman
 Attn: staff
9351 J. J. Hohlfelder
9351 L. D. Posey
9352 G. J. Scrivner
 Attn: staff
9352 C. R. Drumm
9352 W. C. Fan
9352 S. B. Roeske
9352 C. N. Vittitoe
9510 C. C. Hartwigsen
9520 J. W. Kane
9560 J. R. Kelsey

3145 Document Processing, DOE/OSTI (8)
3141 S. A. Landenberger (5)
3151 G. C. Claycomb (3)
8523-2 Central Technical Files