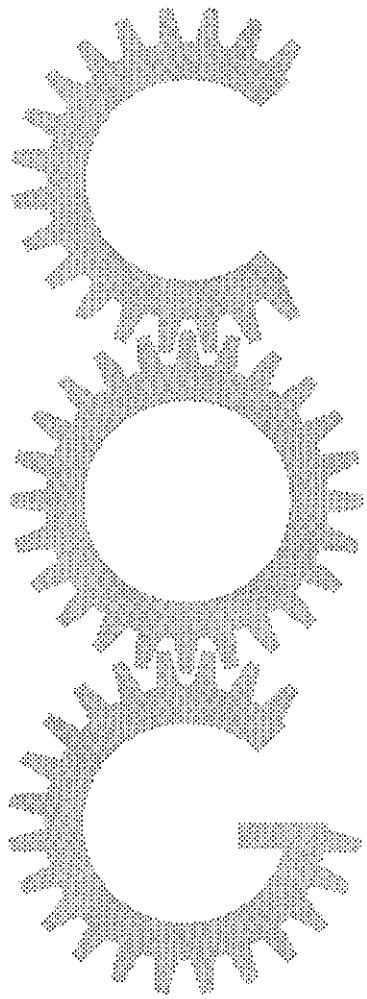
2

M--221-1

DE89 012300



COG –

A Particle Transport Code Designed to Solve the Boltzmann Equation for Deep-Penetration (Shielding) Problems

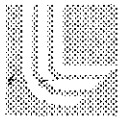
Volume 1

User Manual

1 February 1989

Code written by: Thomas P. Wilcox, Jr.
Edward M. Lent

Manual written by: Thomas P. Wilcox, Jr.



DISCLAIMER

Contents

Contents

Contents

Contents

Introduction

COG is a Monte Carlo computer code designed to solve the Boltzmann equation for transporting neutrons and photons and, in future versions, charged particles. The techniques included in the code make it most suited for solving deep penetration (shielding) problems, but provision is also included for solving criticality problems. The current versions of COG run on Cray-1 and Cray/X-MP computers. We expect that future versions of the code will function on other machines but will still require substantially the same inputs described in this manual.

Summary of Features

Briefly, the features in COG include:

- Cross-section information is described by the point data included on the Lawrence Livermore National Laboratory ENDL libraries.[1] This information is comparable to the ENDF/B data.[2] The treatment within the code uses the fits as described by the input data for both the partial and the differential cross sections. The current neutron library has data from 20 MeV to thermal with the thermal scattering described by the heavy-gas model, except where ENDF/B $S(\alpha, \beta)$ information is available. Photon cross-sections are available from 100 MeV to 100 eV and include the appropriate form-factor information.

- Geometric descriptions are provided by using either analytic surfaces that may go to the fourth order or by using pseudosurfaces describing items like boxes, finite cylinders, and topographic surfaces. A part of the geometry may be defined as a separate unit and then used in one or more places within the problem description. If desired, the geometry may simulate time dependency if you specify a series of descriptions, each of which is used during a given time step. To check the geometry, you may perform volume calculations and draw either cross-sectional or perspective pictures. The latter may be done in color with the appropriate shadows.

- The routines that simulate the fixed source are very general and should allow you to define almost any physical source correctly. For the few cases where the standard inputs are not adequate or are not convenient to use, COG will allow you to input a

previously generated source described on an input disk file. A special option will take a generated source particle and force it to collide in one, two, or three specified regions before allowing it to start its random walk.

- All the normal techniques to modify the random walk are available. These include:

  — Splitting and Russian roulette, both at collision sites and boundaries.
  — Path stretching.
  — Survival.
  — Scattered energy bias.
  — Scattered direction bias.
  — Forced collisions.
  — Weight control.
  — Secondary production control.

- A summary of random-walk events tabulated by regions is given to the user. This includes events by reaction type, boundary crossings, and energy deposition. If requested, you may make plots indicating the location of events in phase space.

- Results may be obtained using estimators based on particle reactions, boundary crossings, or predicted point methods. Additional output is also available for pulse detection schemes. All estimators may include masks covering some part of energy, time, or angle space; in addition, the estimators will include detector responses as a function of these same variables. Besides the usual detector response and assorted statistics, it is possible to request differential response values dependent upon time, energy, polar angle, and azimuth angle, or any combination of these. You may also request results that indicate the importance values associated with events that give rise to responses in a given detector.

- Inputs are as *user friendly* as possible. Data are supplied to the code in a special free-form format, and extensive checking of the input is performed before problem calculations commence. Many input values are presented back to the user by graphic representations, so you can identify input errors more easily.

Comparisons with Other Codes

COG is similar to a number of other computer codes used in the shielding community.. These would include the Livermore codes TART[3]; ALICE[4]; the Los Alamos National Laboratory (LANL) code MCNP[5]; the Oak Ridge National Laboratory (ORNL) codes O5R ,[6] O6R,[7] and MORSE[8,9]; the SACLAY code TRIPOLI[10]; and the MAGI code SAM.[11]

Each code is a little different from COG and from one another. All, except MCNP and some versions of MORSE, only include geometric surfaces up to second order. MORSE, excluding some newer versions, is a group cross-section code that represents cross sections as the average over a given energy range. Scattering is between groups with the angular data represented by Legendre expansions. There is only one averaged velocity per group. TART and ALICE represent the cross sections by a structure similar to group-wise data, but the scattering is calculated by physical models or equal probability bins. Particles keep their individual energies in these codes. MCNP has point-wise cross sections, but it retains the equal probability bins when computing scattered energy. O5R, O6R, and SAM were written some 20 years ago and used methods very exact for the time. Today there are much better physical data available, but they do not fit the forms used by these older codes.

When COG was first envisioned about six years ago, there was one principal consideration—the calculation was to be as accurate as the input data given to the code. Since cross-section data are now presented by the evaluators as point-wise data (i.e., a series of energy and cross-section points with the understanding that interpolation between adjacent points will produce results as good as the data), COG was written to use this form of information directly. The angular scattering data are likewise presented and used as the evaluators present them. No approximation has been made that would compromise the accuracy of these data.

We used every technique we know to run fast on the computer; but we did not used shortcuts, which would produce inaccurate results at some unknown time in the future. We might state that such shortcuts are employed in other codes—they are faster to compute; and in the problems the code writers were thinking of at the time, they did not produce meaningful errors. Users, however, tend to push an existing code into realms where the code

writers did not ever envision it to be used. The result is an incorrect calculation without evidence that it is incorrect.

In the same way as cross-section information, the geometric description of a problem should be as exact as possible. For years Monte Carlo users thought that the entire universe was made up of only figures formed by planes, spheres, cylinders, and, occasionally, by cones. COG gives a much larger list of available surfaces and figures so that the *real* world can be more adequately described. It is probably too much to expect that the old-timers will change and use any of these newer surfaces, but we hope newer members of the shielding community will no longer be restrained by the old ways. The new surfaces take longer to compute; but they also make possible more accurate representations and, in many cases, take much less of a user's time when describing a complex geometry.

Information From Our Experience

This write-up is much less formal than the write-ups we released in the past. We added comments to the sections that describe inputs not commonly used. We did this to explain why anyone would ever want to use these sections of the code. In a number of places we tried to list quickly items that the you should be thinking about, so you will try to avoid making mistakes. In a sense, we are trying to pass on some of our experience to new users. This is the sort of information we have given to those who come to our office and asked questions about solving their specific problems. This is not complete enough to form a textbook but just enough to get people starting to think about what they are doing. There are only a few references listed in this report, however there should be no problem in finding additional material through the usual library sources.

Now let us pass on one very important, and often forgotten, point—you run COG, or any general Monte Carlo code, only as a last resort. If you can get an answer with any simpler method, by all means do so. This would include hand calculations on the back of an envelope, simple point-kernel techniques, quick codes using these techniques, the discrete ordinates codes in one- or two-dimensions, or extrapolation from experimental results.[12-14] However, when you run COG, and run it correctly, you will have the best answer possible. This will be a benchmark calculation for a given configuration or for a new untried system. These results may well be the basis for knowing if a hand calculation would have

worked and will guide you in picking the proper simple methods for use in the future. But COG is expensive to set up and expensive to run. Use it only if you have to.

COG is not a *black box*. Just putting the correct geometric description into the code along with the proper source specification will not necessarily yield the correct detector response as an output. If you do not know what you are doing and if you do not have some understanding of the physics of particle transport, stop and talk to those who do know these things and get their assistance. Years ago an experienced shielding analyst passed on some advice to me. It went something like this—"Monte Carlo is a marvelous technique. You can set a problem up spending weeks of time doing so and then it can run on a computer for many, many hours. When you get an answer, and one with a very small indicated standard deviation, only then will you realize that it is three orders of magnitude smaller than the experimental result." We hope to show you how to avoid such pitfalls, but you cannot do so without paying a lot of attention to every part of your problem input and to all the information that COG will give to you.

How to Use the COG Manuals

This is the first of a number of volumes that will be written about COG. The first three are entitled *Users' Manuals,* and are designed to assist a code user with the problems associated with getting answers from the code.

This users' manual is Volume 1; it contains a full description of the required inputs and what they mean along with a very simple problem. Volume 2 will contain more difficult problems and will illustrate the inputs described in Volume 1. Volume 3 will explain how you run a problem effectively. Volume 4 is available and describes various benchmark problems. Volumes beyond this will describe in detail what is in the code and how to generate your own cross-section files and other information that is of interest to those who write Monte Carlo codes.

In reading this volume, we suggest that you go to the next chapter, **Getting Started** (page 7), and read at least the sections **An Overall View of a Problem Specification**, **Free-Form Inputs**, and **Terminology Used to Specify Inputs** on pages 10, 18, and 19 to obtain an over-all view of a problem specification and the terminology used throughout the remainder of the manual. After you get to the point of running your own problem, you can come

back and read about obtaining the code and how to understand the error comments it will produce. Then go on to the chapters **Overall Problem Parameters , Defining the Geometry, Specifying the Fixed Source, The Random Walk,** and **Specific Results** (pages 25, 33, 109, 145, 167, respectively)—and read the first section of each. This will give you a greater perspective of problem specifications without getting into very detailed descriptions. Then read those things that are of interest to you, ending with the example problem in the chapter, **Putting It All Together—An Example Problem** (page 203).

Getting Started

Obtaining and Running the Code

COG and all needed cross section libraries are located in the Octopus time-sharing system public files on all Cray computers at the Livermore Computing Center (LCC) of LLNL. The following instructions are for using the code at LCC. A version of COG is also available at the National Magnetic Fusion Energy Computing Center (NMFECC) at LLNL. The availability of the code at other locations is dependent upon effort expended at those locations, and the appropriate running instructions will have to be issued as part of that effort.

To run on the Octopus time-sharing system, log in and type:

```
cog input-file output debug box  / time value
```

where the four options following the code name `cog` may be given in any order or may be omitted. The user-defined information on the input line is in *italic typewriter letters*; the program-defined information you type on the input line is in `regular typewriter` letters. The four options on the above input line are defined as follows:

input-file — Name of the file containing an ASCII description of one problem. The default name is `COGINPUT`.

output — Normal code output is directed to the microfiche camera. If desired, the output files will be saved as DLI files by including one of the following words on the input line: `keep`, `k`, `save`, or `s`. The utility routine DLTV may then be used to direct the output to one of the other available output devices.

`debug` — The inclusion of the word `debug` will cause the code to keep the output files and to exit when all the input has been read without following any random walk particles.

box	The appropriate box number to which output is to be directed (i.e., S55). If omitted, the code will use the box number that the system associates with the current user number. The word *box* itself is not input.

The *time* and *value* inputs are the usual computer time you are requesting and the amount taken from your bank account to cover the requested time. The priority is their ratio.

After the code has read all available input and cross-section data and has started following particles in the random walk portion of the calculation, typing the word where on the associated terminal will cause COG to inform the user which source particle is being followed. Typing end will cause termination of the problem after the current source particle has been completed. All normal code results will be calculated and output. Providing these inputs before the code has reached the random walk section of the code may, or may not, result in the above actions.

For night and weekend runs, jobs may be submitted using the COSMOS system. The following example illustrates one way that will work using COSMOS. Other ways may be used—consult the COSMOS write-ups to develop them.[12]

Assume, for purposes of this illustration, that your user number is 123456. Perform each step in the order given below:

1. Log onto a Cray computer and enter XPORT to create a directory listing in the take (.t) directory. Execute the XPORT instruction `create .t:cprob`.

2. Use TRIX AC to generate your problem input in a file that, for example, you call prob1. Store this file in the above .t:cprob directory with the XPORT command `write .t:cprob:prob1`.

3. Again use TRIX AC to generate a file you name, for example, runfile. Assume that your box number is S55 and that the

COG problem is to run for 18 minutes. Put the following information into file `runfile`:

```
*select box=s55, tasktimelimit=18.
*interrupt on timeup to lab1
*xport /rd .123456:cprob:prob1 \lf\end
*cog s55 k prob1
*go to lab2
*lab1:
*restart end
*lab2:
*destroy coggxs coglex
*give alwith. d0- 999999 k. end
*xport /r t. .123456:cprob [alwith. cog d0-] p. t\lf\end
```

These instructions will read the file `prob1` into the computer, start COG and allow it to run for 18 minutes. If the problem has not terminated on its own within this time limit, the END command will be issued to it to force termination. Output from the code will be given to the system and, along with all COG generated files, will be stored in the specified `.t` directory.

4. The `runfile` is then given to COSMOS by executing a routine called SUBMIT. Assuming that the total running time should not exceed 20 minutes, the form required to do this is:

   `submit i=runfile,t=20, end / time value`

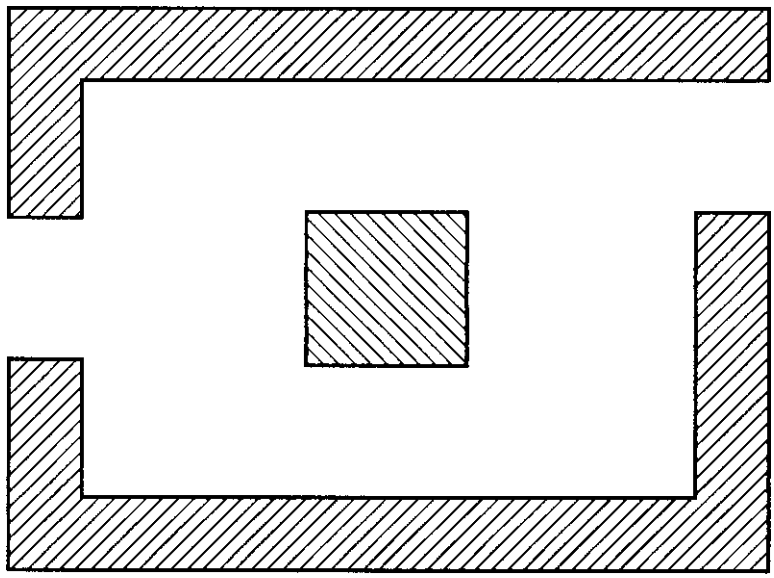
An Overall View of a Problem Specification

The Physical Problem

Often code write-ups assume that you already know everything that goes into the code and then present to you only the fine details of the input. For those who are not experienced with COG or similar codes, we will give a very quick overview of the information that the user must provide. We will then broadly show how COG expects this information to be presented to it.

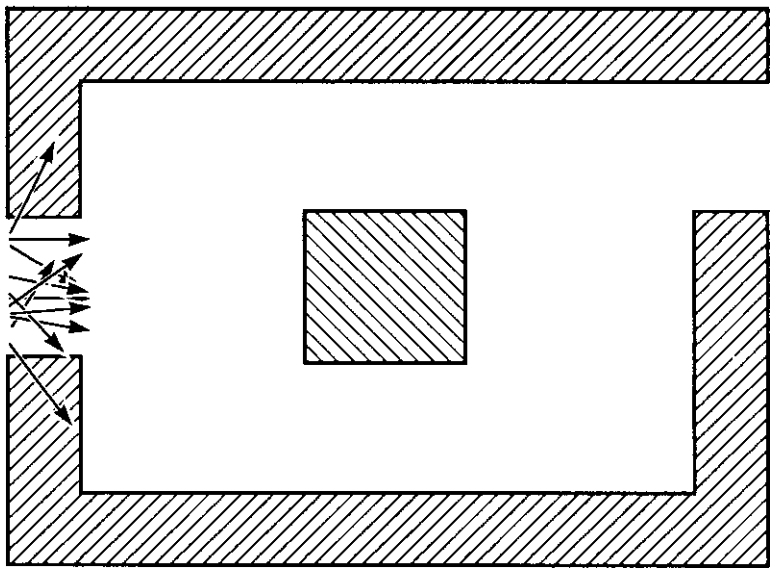
To transport particles, you must provide a geometric description of the *things* in that part of the universe in which your particles are going to move. This description must include physical boundaries to these things as well as the physical description of the materials involved. The material definition, of course, will give rise to cross-

section information affecting the movement of the particles. As an example: assume we have a room with thick walls, a door on the left wall and another on the right, and a big square piece of something in the middle of the room. Neglecting the floor and ceiling, a cross-sectional cut through the middle of the room would look like:

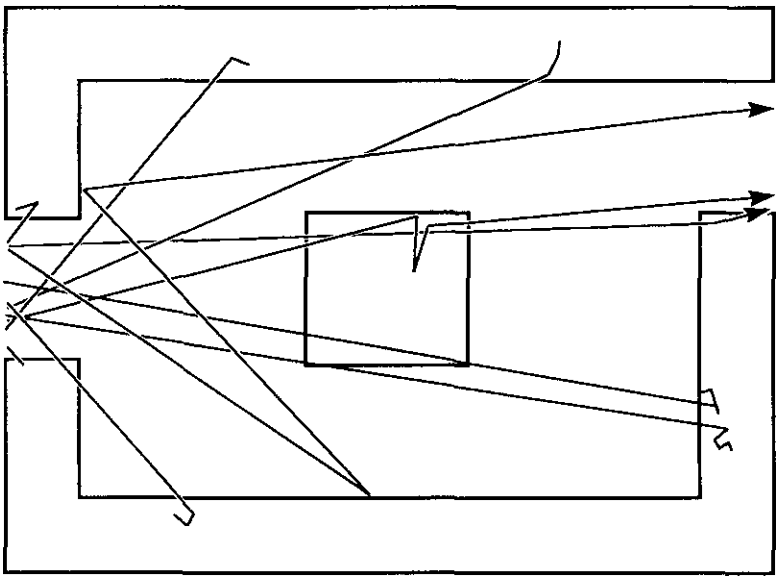


Now that we know where materials are, we must specify the fixed source of the particles we are going to follow. This would include the particles' starting location; their type; their energy; the initial direction they are going; and, for time dependent problems, the time that they start. Obviously, different source specifications will produce different results even though the geometric description is the same. The theory of superposition holds for shielding problems—i.e., if a given source specification produces an answer A, and another source specification produces an answer B, a source specification identical to the sum of the two specifications will produce an answer equal to A + B.

For our illustration, assume we have a source specified at the left-hand door. A few particles from this source would look like:



Each particle will now start from its initial source position and will travel through the defined materials. Somewhere along this initial path, as determined by cross-section data and a random choice, a collision will result. In most cases, a particle will come out of this collision with a changed energy and going in a different direction. At times, more than one particle will emerge; and with problems involving a number of particle types, a different particle may also emerge. Each particle will be followed until it is either absorbed in a collision or it leaves the defined geometry. This is the same process that Mother Nature follows except that she calculates faster than we do. Traditionally, this is referred to as *analog* Monte Carlo. For our example, we now indicate some of these particle trajectories.



Now the only thing missing is an answer. Under natural conditions, the moving particles cause something to happen to the materials in which they interact. This could be heating, or activation, or cell damage, or related type of events. An experimental physicist would place some type of detector into the real world geometry, and it would produce a numerical result. In performing the calculations, we will also put something we call a detector into the problem, and it, also, will produce a numerical result. It is easier to do this in a calculation, since we have all the parameters of the particle (energy, position, time, and direction) available to understand what is going on. In our example problem, a detector could be simulated by standing by the right-hand door and counting particles leaving the room.

Input Blocks of Data

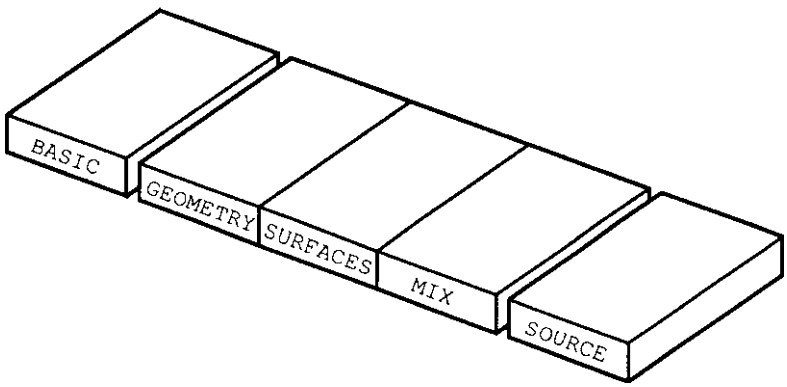
The input to COG is formed of a number of *blocks* of input information. Each of these blocks starts with a block name and is followed by input data. The block normally will be terminated by the start of another block but may also be terminated by the word `last`. Needless to say, the names of the blocks are *reserved* words and should not be used within the problem except to denote the start of blocks or hidden away within specified titles. There are many possible blocks of data. Not all of them need to be provided in any one problem. In general, if the function of a block is not wanted in a given problem, just leave the block out of the problem definition.

Each problem starts with one record that will be used as a title throughout the problem output. Like all input records, this may be up to 80 columns in length. This is then followed by the blocks of input information necessary to describe the problem. These may be given in any order. The conclusion of the problem input is denoted either by the single entry `end`, which is treated like a block, or by the end of the input file. All data after the word `end` will be disregarded, thus making a convenient place to store input that was not used in the current problem run but that was previously used or will be used in subsequent runs.

Five blocks are *required* in all problems. These provide the necessary information to describe a problem as discussed previously. Included in this group are:

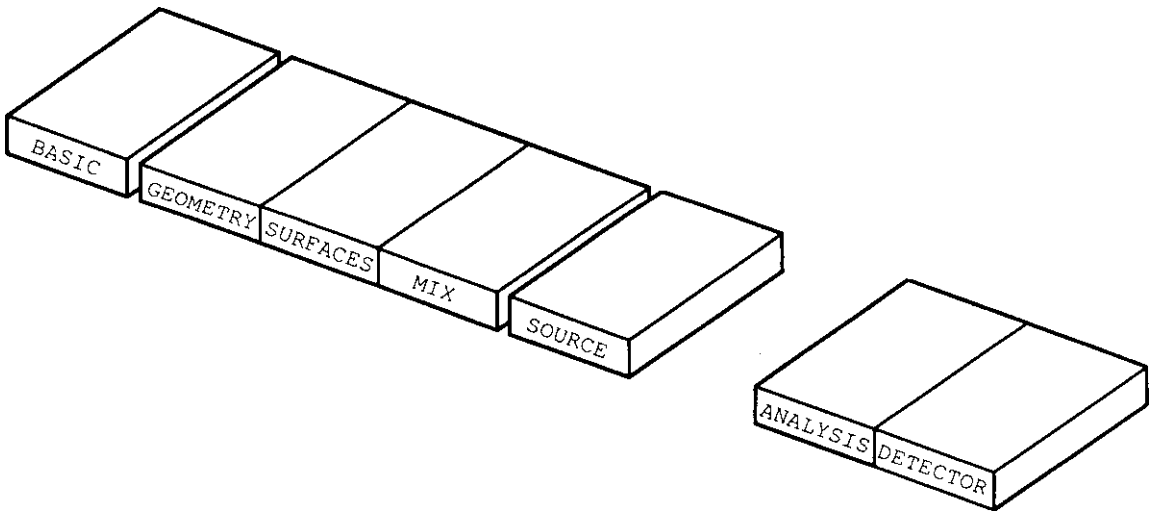
BASIC	Particle types to be followed along with other basic problem information.
GEOMETRY	Definition of the geometry of the problem.
SURFACES	Surface definitions used with above.
MIX	Mixture compositions.
SOURCE	The starting source.

If we illustrate these blocks of input information as physical blocks, we now have:

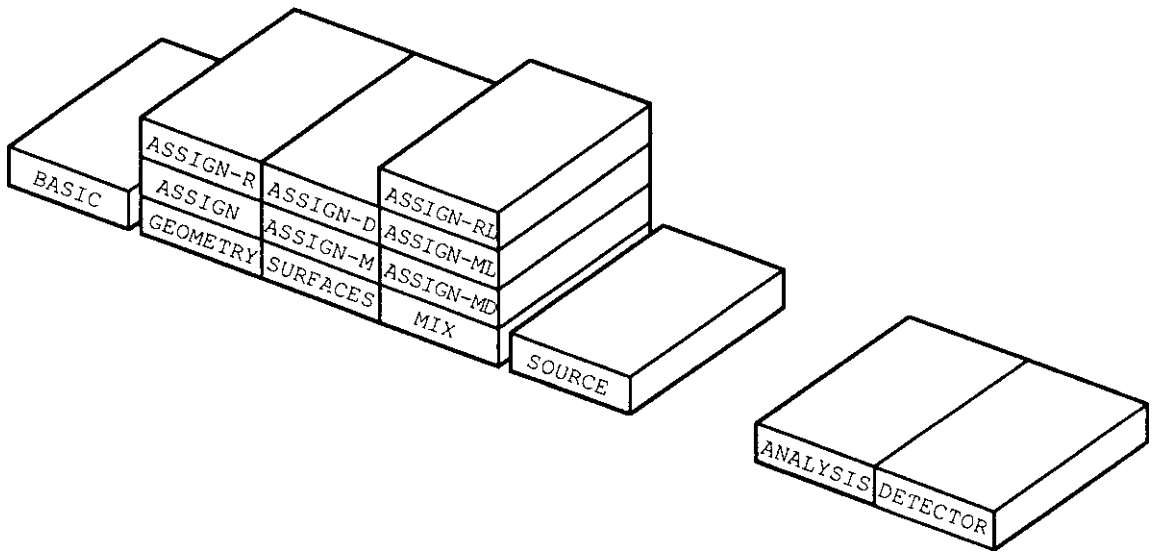


Now if we ran a problem with just these five blocks, we would get some basic results—energy deposition and reactions by type in specified geometric regions as well as how many particles crossed boundaries. If we want more data we would add the following two blocks.

DETECTOR	To obtain real results at specified locations.
ANALYSIS	To see where events are happening in phase space.

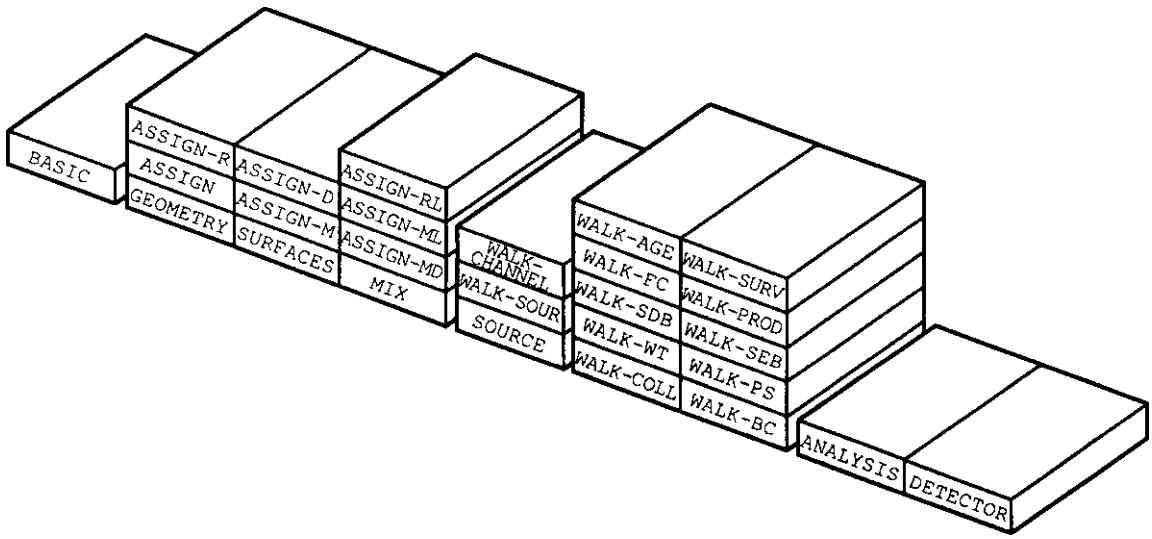


To make the combination of geometric position and mixture number easier and also to make the places to look at what happens in the random walk easier, we would add one or more of the ASSIGN blocks:



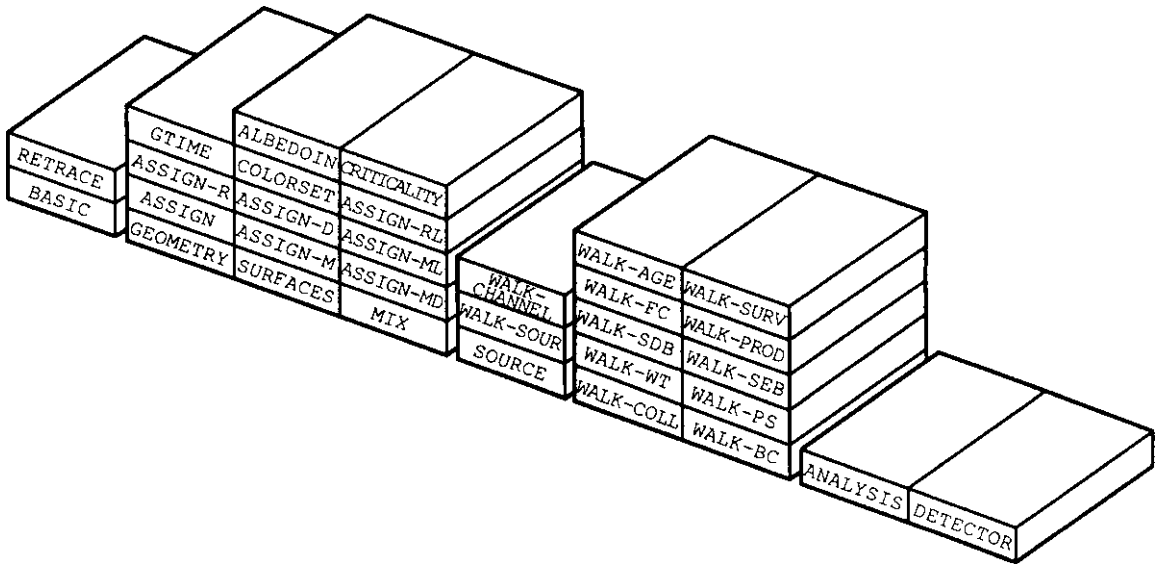
Getting Started

To be more efficient in following the paths that yield an answer to our specific problem, deep penetration problems normally require that we vary the random walk. There are many techniques to do this. Inputs for each of these are included in the series of blocks that start with the name WALK:



Finally, there are five blocks that do very special things.

RETRACE	Allows the reconstruction of the random walk of a given particle.
ALBEDOIN	Reads an ALBEDO definition.
COLORSET	Defines special colors for pictures.
GTIME	Activates the time-dependent geometry.
CRITICALITY	Replaces the SOURCE block for static multiplication calculations.



Free-Form Inputs

Input to the COG code makes use of a free-form input designed specifically for this code. Input values can be positioned according to the desires of the user. Experience indicates that each user has his own idea of what forms a neat and logical input. It is suggested that you develop your own input format—several will be illustrated in the example problems presented in Volumes 1, 2, and 3 of this series of reports.

Inputs are separated by blank spaces, commas, equal signs, or the end of a line of input. All four of these are equivalent, and any use of any combination of them in any number is still equivalent to only one separation. Throughout this write-up, we used which ever separator seemed to convey the clearest idea of the input.

Any input that starts with an integer, a plus sign, a minus sign, or a decimal point will be considered as a numeric value. Any of the following numeric inputs will convey the value of one thousand to the code:

```
1000 .1000.  +1000.  +1000.000  1.0E+03  1.0e+3
1.0E+3  1.E+3  1.0+3  1+3.
```

Numeric inputs that represent integer values may be input as floating point values. An input of `27Z` will result in twenty-seven zeros being read by the code, while an input of `27R1.63`, the `R` signifying REPEAT, will result in twenty-seven values of 1.63 being read. An INTERPOLATE option will allow an input that looks like `1.0 9I10.0` to give 1.0 2.0 3.0 4.0 .... 9.0 10.0. The use of the letter `L`, rather than `I`, will produce a logarithmic interpolation. The REPEAT PATTERN option allows the specification `1 2 3 4RP3` to yield 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3—i.e., take the previous three input values and insert them again four times. A specification of `77B` indicates that the input value is in octal.

Inputs that are not numeric will be considered to be alphabetic. In general, these are key-words that inform COG of your desires. Within the code, only the first eight characters of such inputs are retained, but the input may specify as many characters as you desire. In a number of cases, concepts are expressed as two words in the input. These are written with a dash between them, so the code recognizes only one input. The dash is important in these cases, or COG will think there is one more input than required. An example of this would be a detector definition of

BOUNDARY-CROSSING. Capital letters are equivalent to their lowercase counterparts and may be employed to clarify the input.

An input enclosed within quotation marks will be considered to be a binary-coded decimal (BCD) string of data. This will be conveyed to the code as written complete with capital letters and spaces. Such a string must be specified on a single line of input thus limiting it to 78 characters in length. COG uses such specifications for additional titles to be used with the code output.

Comments may be imbedded within an input deck by using the dollar sign ($). Everything after a dollar sign and before a second dollar sign or the end of the input line, which ever comes first, will be considered to be a comment. It will be printed out when the input is first listed at the beginning of a COG problem and then discarded. This forms a nice way to write yourself notes explaining where certain information came from, what it was you were doing, and approximations that were made. When you go back to rerun a problem that was initially run months before, such comments can be very valuable.

Terminology Used to Specify Inputs

The majority of the rest of this report presents input formats that enable COG to perform desired calculations. Rather than have to define the meaning of each of these formats every time they occur, we will present the underlying syntax here.

- Alphabetic input words or symbols are indicated by uppercase letters (i.e., SURFACES). The actual input to the code may use either upper or lower-case letters.

- Numeric inputs are indicated by lower case descriptions. Dashes are used in these descriptions to clearly indicate that only one number is to be input (i.e., inner-radius).

- Required inputs are those that are not enclosed in square brackets, parentheses, or curved brackets. These inputs must be provided.

- One of the items of a set enclosed in square brackets and written under each other is required and must be provided. In most cases, these are optional input words which accomplish the same thing. As an example:

```
[SPHERE]
 SPH
 S
```

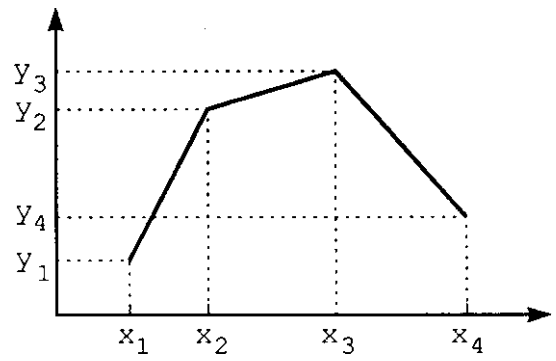
  A new user will choose `SPHERE` so that he can make sense out of his input listing. After some experience, he will choose `SPH`; and with a lot of experience, the letter `S` will be sufficient. *The square brackets are not included.*

  If the options within the square brackets represent different options to the code, rather than just variations of input words, any data indicated after the key-word are also part of the required input.

- For input for which you have already provided some items and then encounter one or more pieces of data enclosed in curled brackets or braces, { }, these additional items are optional. They may or may not be provided depending upon what it is you wish to accomplish. *Do not include the curled brackets or braces.*

- In a similar case where data are enclosed in parentheses, (), these additional data are optional but, if provided, must have been preceded by previously requested data that also were enclosed in parentheses. *Do not include the parentheses*

- Throughout the code, many inputs require that a curve be specified. This may be done by providing a straight-line approximation between given points as follows:



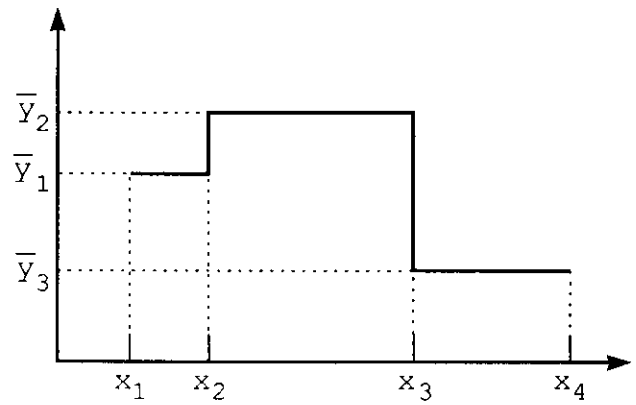
where the code inputs would be:

x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4

The fit assumes that y is zero for all values of x less that the minimum provided and that y is also zero for all values of x greater than the maximum provided. The x values may be either increasing or decreasing in the input. This type of input is represented throughout the report as:

$(x, y)_i$

A curve may also be specified as a set of bin information:



with the actual inputs being:

$$x_1\ \bar{y}_1 \qquad x_2\ \bar{y}_2 \qquad x_3\ \bar{y}_3 \qquad x_4$$

Again y is zero at x values less than the minimum value of x provided and at points greater than the maximum value of x provided. The values of x may be either increasing or decreasing. Note that one more x value than y value is input. This input will be represented by

$$(x, \bar{y}\ )_i$$

- There are a number of items that occur as choices throughout an entire section of this report. Rather than repeat the specified input time after time, we have taken the shortcut of just shortening the specification to some identifying part followed by a series of dots. Thus, you would input the translate/rotate (TR) command as,

  (TR x_0 y_0 z_0) (x_1 y_1 z_1) (x_2 y_2 z_2)

  However, after it has been discussed in the text the TR command is represented by:

  (TR ....)

Code-Detected Errors

The first thing COG does is to read all the input of a problem and to convert the free-form values to the appropriate numeric or alphabetic forms. If there is a numeric input that will not convert to a real number or if one of the essential blocks has been omitted, the listing of the input will identify a fatal error and the problem will be terminated. If a block has been repeated, and it should not have been, the code will also indicate a fatal error.

From this point on, COG will take one block at a time, read each input, and print out a full description of what the code thinks it means. In many cases, graphic representations of the inputs are presented to enable the user to verify quickly that the proper data have been input.

The code will check the inputs in as many ways as possible. If they are known to be incorrect, a FATAL MESSAGE will be printed indicating just what is wrong. These error messages have the same form throughout the code and clearly show up in the printed output. If the information is unusual, but not necessarily incorrect, a WARNING is printed. In both cases, the code will then go ahead and try and read as much of the remainder of the input data as possible. However, if a fatal input error was present, no attempt will be made to enter the random walk phase of code operation.

There are two problems with error detection methods in checking code inputs. The first is that the code writer did not think of all the errors that users are capable of generating. This means that some incorrect data may get through and cause problems later in the random walk section of the code.

The second is related to the coding that detects the errors. Frequently, something is wrong with that part of the program, and a fatal operational error occurs that masks the real problem. It is just not possible to check the code by running with every conceivable input error, much less with combinations of every conceivable error.

When you are trying to run a problem and the code blows-up during the input phase, look to see if you made an error. Fix it, and try again. Most of the time this will solve your problem, and you can go on with your work. Then come and tell the COG programming group what happened, so we can fix the code.

Overall Problem Parameters

There are a few parameters that set the basic description of a COG problem. These enter into the required input information used in many different blocks of information or form limits to the problem's operation. All are input in the BASIC input block of information and may appear in any order.

Basic Problem Parameters

Particle Types to be Followed

particle-type-to-be-followed is the only variable information that *must* be supplied in this block. All other data have default values inserted by the code.

Simply, *particle-type-to-be-followed* supplies the names of one or more of the particle types that are to be followed in the problem. Acceptable names include:

```
NEUTRON
PROTON
DEUTERON
TRITON
HE-3
ALPHA
PHOTON or GAMMA or GAMMA-RAY or X-RAY
ELECTRON or BETA or BETA-
POSITRON or BETA+
```

A problem in which only neutrons are to be considered would include the single name NEUTRON. If gammas produced from neutron reactions are also to be followed, the input would contain both the names NEUTRON and PHOTON.

In the energy balances performed within the code, energy carried off by a particle type not being followed will be deposited at the collision site.

Version 1 of COG can follow only neutrons and photons. Requests for particle types other than these will result in an error message. The acceptable ranges of energies for each particle type is determined by the associated library of cross-section information.

Overall Problem Parameters

Units of Distance, Energy, and Time

In most Monte Carlo codes there are fixed standards for the input units to be used in the problem specifications. Frequently, misunderstandings occur as to what these definitions are, and these misunderstandings then give rise to problem errors. On other occasions, the user spends a lot of time converting dimensions to the standard units. COG allows the user to specify the input standards.

Unless you specify differently, COG uses the units CENTIMETERS, MEV, and SECONDS. The units are indicated in the printout of the problem input-description to remind you of the units used.

To change the standard unit of length, one of the following may be added to the BASIC block inputs:

```
ANGSTROM  or  AU  or  A
MILLIMICRON
MICRON
MIL  or  MILS
MILLIMETER  or  MM
CENTIMETER  or  CM
INCH  or  INCHES  or  IN
PALM
DECIMETER  or  DECIM  or  DM
HAND
LINK
FOOT  or  FEET  or  FT
CUBIT
PACE
YARD  or  YD
METER  or  M
ELL
FATHOM
ROD
ROPE
DEKAMETER  or  DEKAM
CHAIN-GUNTHER'S
CHAIN-RAMDEN'S
HECTOMETER  or  HECTOM
FURLONG
CABLE
KILOMETER  or  KM
MILE-STATUTE
MILE-INTERNATIONAL-NAUTICAL
MILE-ENGLISH-NAUTICAL
```

```
LEAGUE-STATUTE
LEAGUE-INTERNATIONAL-NAUTICAL
LEAGUE-ENGLISH-NAUTICAL
ASTRONOMICAL-UNIT
LIGHT-YEAR
PARSEC
```

Note that this changes the unit of length throughout the problem—thus, if the standard of length is set to be a meter, the unit of area in a detector definition must be given in square meters.

Round-off errors in machine computations can cause problems in the geometric treatment of surfaces in any Monte Carlo code. For this reason, it is suggested that the units used go along with the size of the geometry being considered—i.e., if transport is being calculated within foils with dimensions of a few mils, the standard of length should be in mils.

The standard unit of energy may be set to any of the following by adding the appropriate name to the BASIC block:

```
MEV
KEV
EV
JOULE
ERG
JERK
```

Likewise, the standard units of time may be chosen from the following:

```
NANOSECOND  or  NS
SHAKE  or  SH
MICROSECOND
MILLISECOND  or  MS
SECOND  or  SEC  or  S
MINUTE  or  MIN
HOUR  or  HR  or  H
DAY  or  D
MONTH  or  MO
YEAR  or  YR
```

Overall Problem Parameters

Maximum Running Time

Normally, a user will specify that a problem is to run a fixed number of source particles. When all these have been followed and the associated results obtained for them, the run will end, and final results will be output to the user. On occasions when users do not have a good feeling for the running time per source particle, they may wish to protect themselves by setting a maximum time the problem is to run before entering the output phase. This may be accomplished by adding to the BASIC block data of the format:

```
TMAX = time
```

where *time* is the specified time in minutes. If not provided, the maximum time will be set to 1.0E+8 minutes, which effectively eliminates use of this option.

A problem may also be terminated by sending the END message to the running code as described in the chapter **Getting Started**, section **Obtaining and Running the Code**, page 7.

Starting Random Number

Monte Carlo codes use sequences of random numbers to determine what will happen to a particle in the random walk. These are really not random sequences but, instead, are pseudorandom-number sequences that start with a given seed value and use an algorithm to obtain the next value of the sequence. If the starting seed is the same for two sequences, each number in the two sequences will be the same. In simple terms for the COG user, if two identical problems are run with the same initial starting seed, both will produce identical results. This is not normally desirable so the COG code starts each problem with a seed derived from the microsecond clock. For all purposes, the initial seed is therefore randomly chosen. This seed is printed in the code output.

On rare occasions, like debugging, it is desirable to duplicate a random-number sequence. To do this, the following information is added to the BASIC block:

```
RN = number
```

where *number* is generally taken from a previously run problem and, to obtain the needed accuracy, has an octal format.

Special Problem Controls

Follow Two Photons in Pair-Production Reactions

In cases where charged particles are not being followed and when a photon initiates a pair-production reaction, two 0.51-MeV photons are emitted in opposite directions from each other. The same results are obtained in almost all Monte Carlo calculations if only one photon is generated and followed provided, of course, that that photon has twice the normal statistical weight. There are some special cases when it is necessary to follow each new photon separately. These cases generally fall into the category of coincidence, or anticoincidence, detectors (see the description of such detector use starting on page 175).

To force COG to generate both of the pair-production photons, the following statement is included in the BASIC block:

```
PAIR-PRODUCTION
```

Include Delayed Neutrons in Fission Reactions

The fission process produces neutrons that are effectively released instantaneously and a few that are delayed from seconds to minutes after the initial event. Normally, only the instantaneous neutrons need be followed. In problems involving steady-state reactor operation, the delayed neutrons should also be included. To accomplish this, include the statement:

```
DELAYED-NEUTRONS
```

in the BASIC block of data. The time delay of these neutrons is presently not simulated by the code, thus implying that this option is only correct for steady-state problems.

Overall Problem Parameters

Importance Weighting with Point Estimators

Including the input:

```
PFE-IMPORTANCE
PFE-I
PFE
```

turns *off* the automatic importance weighting of choices made from the scattered energy distributions when doing point flux estimations. If you do not include this input, an approximate importance function is calculated within the code using a built-in few-point energy grid folded with the energy grid of the detector response function. For those reactions that have an energy distribution of emergent particles, an importance proportional to the response function times the attenuation from the scattering point to the detector is calculated on this grid and employed.

Fission-Produced Neutrons and Photons

If the entry:

```
NOFISSION
```

is included in the BASIC input block, no neutrons or photons will be produced in a fission event. This option is essential in a critical or supercritical system when the original fixed source includes these particles.

Storage of Code-Generated Files

COG produces a number of disk files during the execution of a single problem. Each of these has a name of the form:

```
COGxabc0
```

where *x* is a single letter identifying the type of information in the file, *abc* represents three letters randomly chosen by the code to identify a specific problem, and the ending number or letter identifies the position of a file in a sequence of files of the same type.

The types of information presently written and represented by *x* are:

S	Initial source information.
A	Data obtained in the analysis of the random walk.
D	Results for all detectors.
R	Red output for color pictures.
G	Green output for color pictures.
B	Blue output for color pictures.

These files may be left at the end of a problem run. However, because these files may be needed by the user again, provision should be made to save them, especially in night runs.

Early versions of COG employed an input in the BASIC block to accomplish this storage. Experience indicated that users preferred to employ the options in COSMOS for this function. Therefore this option has been removed from the current COG program.

Overall Problem Parameters

Defining the Geometry

General Outline of Geometry Specifications

Sectors, Surfaces, Mixtures, and Assignments

To relatively new Monte Carlo users, the specification of inputs to describe the geometry of a problem is usually a major obstacle to be overcome before a solution to the problem is obtained. In many cases it becomes such an obstacle that most of a user's available time is spent describing the geometry and so little time is left that he never understands if a correct solution to the problem was obtained. More seasoned users approach the geometry with a little more understanding but still with a great deal of fear.

COG was written with the user in mind, and the code inputs were designed to make this task as easy as possible. However, just stop for a minute and look up at the room around you. Now try to put in words, in any way you want, the physical description of the items around you. Just from the large number of things in the normal office, this is no simple undertaking. A large part of the geometry game is to understand radiation transport well enough to know what to describe in exact detail and what can be simplified.

Many incorrect calculated results are obtained because the user oversimplified the existing geometry. Some items are omitted from the problem, while others are approximated as *iron* rather than the complex *stainless steel* composition that was really present.

A number of years ago, a group reported on its calculations of a shielding experiment. Try as they would, they could not reproduce the experimental results. Finally, they looked again at the physical configuration they were modeling. Very evident in the experiment, but missing in the calculation, was an aluminum ring holding various parts together. This had been left out of the geometric model, since everyone knows that aluminum does not cause much attenuation for neutrons. However, it does scatter neutrons. When the aluminum ring was included in the problem description, the calculated results looked like those obtained experimentally.

Defining the Geometry

After setting up a problem, it does not hurt for the users of COG to go and physically look at the geometry they are modeling. When doing so, they should always remember that neutrons and photons easily pass through materials that light will not transverse.

At the other extreme, too much time can be spent putting in unnecessary detail. Experience will teach you when items can be lumped together or fine detail omitted. Generally, regions a long way from the point at which an answer is desired can be simplified, unless particles pass through that location on their way from the source. Those users lacking experience may find it worthwhile to run a number of simple problems to evaluate the effects of problem detail.

When you start to describe the geometric configuration of your problem to the code, you will need to make four lists. These will be, effectively, the input for four different blocks of data.

First, you will have a list of different surfaces that form the boundaries between the different *things* you are describing. For convenience, you will assign numbers to these surfaces, and each will have some definition of surface type, size, and location. The section **Surface Definitions** in this chapter (page 37) will give all the fine details you will need to make these definitions in the SURFACE block of input.

Second, you will define a piece of volume that the code refers to as a *sector*. You will do this by reference to the surfaces just defined. You will assign a number to each sector to identify it to yourself and to the code. Some of these sectors may signal an outer boundary of the problem or that a more detailed problem description is contained therein. These sector definitions, along with ways of checking to insure the descriptions are correct, are included in the GEOMETRY block of input and are described in the section **Sectors and Related Items** starting on page 77.

Third,there will be list of the different materials used in the problem. Materials are something with which we are all familiar. They are things as steel, water, sand, or air. The MIX block of data gives the exact composition of each material used in the problem and associates each material with a material number. The section **Defining the Material Media** contains the full explanation of this block (see page 102).

Statistical regions are an artifact dreamed up so the the user can attach to a given geometric part of the problem a set of parameters

that alter the random walk of particles and, hopefully, make the solution to the problem more efficient. The region numbers also separate the problem into geometric parts over which various summations are made to inform the user just what is happening during the random walk. The region numbers are also used in the definition of detectors.

Fourth, the various ASSIGN blocks attach a material number and a region number to each sector number. It is also possible to specify a relative density factor to be applied to the material within the sector. You will find the details of all these inputs in the section **Tie the Geometry Input to the Rest of the Problem** starting on page 98.

It should be pointed out that the sector is the smallest unit of geometry you can specify. In some cases a group of sectors contain the same material but differ in region numbers. In other cases, much of the problem has the same region number but contains different materials. Experience indicates how fine the geometry of any specific problem should be broken up. For simple cases where the sector, region, and material numbers are always equal to each other, the ASSIGN blocks can be left out of the problem.

A Systematic Approach to Geometry Inputs

It seems wise to stop at this point and make some very general comments. Some of the words may not yet have meaning for a new user, but stop and do some thinking and then come back and re-read this section when the geometry inputs are more familiar to you.

After watching many users set up the geometry for complex three-dimensional problems, it is evident that those who have the least amount of difficulty are those who follow a systematic approach to setting up their geometric definitions. Now each can have a different approach—exactly how you do your work does not seem to be important—but it is essential to develop such a systematic approach for yourself. Needless to say, simple problems require little effort, but the real test is a good, big, messy, and complex problem—as all the buildings in New York City. I am going to describe a method that I use—you may modify it to your own needs and desires. The method is only presented to start you thinking.

First, we assume that you have four pieces of paper in front of you. As described above, one is labeled *Surfaces*, one *Sectors*, one *Assignments*, and the last *Mixtures*. Now you will take another

Defining the Geometry

sheet and sketch a cross section of the geometry. If this is a simple case, you may need only one cut through the 3-D geometry to show everything you need. If it is more complex, you may need a number of sectional views or a number of separate sketches. This can be a simple hand sketch, but surface locations should be illustrated with exactness—i.e., a plane falling to the left of a spherical surface should not be drawn to the right of the surface. Now, do the following:

- Pick some surfaces and assign numbers to them. Write these numbers on the sketch either in a given color that means *surface number* or with boxes or circles around them to mean the same thing. Indicate the positive and negative side of the surface, as discussed in **Surface Definitions**, below, by adding plus (+) and minus (–) signs to the sketch.

  On the sheet marked *Surfaces*, write the definition for each surface to which you have just given numbers. Insure that these numbers have not been used before.

- Now define some sectors by their relationship to defined surfaces. Write the definition on the sheet marked *Sectors*, and indicate a sector number on the sketch. Make sure that this number can not be confused with a surface number. Shade in the defined sector on the sketch. The shaded part should not overlap any previously shaded portion. At the end of the geometry definition, all parts of the sketch not shaded will be the default material (normally a void).

- On the sheet marked *Assignment*, write down the sector number and the associated material number; include the statistical region number and any density factor.

- If the used material number is not already on the sheet marked *Mixture*, add it with a definition of this specific mixture.

- Recycle through the above steps until everything is defined.

As you understand more of the geometry inputs available to you, there will be slight variations and additions to the above. Users should be able to make such adjustments for themselves.

Surface Definitions

Numbering Surfaces

Each surface used in a problem is given an identifying number by the user. This is any integer number from one to 9,999,999. Surface numbers do not have to be used consecutively—i.e., you can skip over numbers—neither do the defined surfaces have to be input in increasing order. You may define a surface you do not use. However, if you give two surfaces the same number, a fatal error will occur, and your problem will not go into the random-walk phase.

The surface number itself is used only for identification purposes. You are encouraged to use numbers that have a meaning to you and would help to identify where you are in a problem if an error should occur or if you wish to make changes later. Such a system of numbering becomes more necessary as the complexity of the problem geometry increases.

Positive and Negative Sides of Surfaces

After you give an identifying number to a surface, specify it more exactly by its type (i.e., PLANE or SPHERE) and some specific physical values that tell of its size, etc. No matter what type the surface is, it is going to be represented within the code by one or more analytic definitions of the form:

$$f(x,y,z) = 0$$

Take, for example, a sphere with a center at the origin and a radius R. The surface itself is represented by:

$$f(x,y,z) = x^2 + y^2 + z^2 - R^2 = 0$$

and we note that $f(x,y,z)$ is less than zero for points closer to the origin than R and greater than zero for points further from the origin than R. We then can attach to this surface the concept of a *negative* side and a *positive* side depending only on the evaluation of the analytic surface equation. For many simple surfaces, as the sphere, the negative side turns out to be *inside* and the positive side is *outside*. However for something even as simple as a hyperbolic paraboloid, this concept of inside and outside is not as easily defined.

Defining the Geometry

Two problems occur with this concept. The first is with a plane. This is represented by:

$$f(x,y,z) = c_0 + c_1x + c_2y + c_3z = 0$$

If each coefficient is multiplied by –1, the location of the plane is not altered, but its positive and negative sides are interchanged. Therefore, it is necessary to specify which side is positive with an additional user-supplied piece of information.

The second problem is more complex. It occurs in surfaces with terms of order greater than two. In developing these surfaces, squaring expressions of lower order gives rise to the loss of algebraic signs. The resulting equation may not at all represent what the user had in mind for places in space. An example of this occurs in the degenerate torus in which a point can be logically inside the figure but mathematically outside it. You should carefully check such figures to insure that the your definition and the definition of the code are the same.

COG allows the use of many pseudosurfaces. These surfaces are made up of a number of analytic surfaces. Thus a BOX is a pseudosurface made up of six planes. To be *within* the box, a point must be on the negative side of each of these six planes. The fact that this particular surface type involves this complexity is built into the code itself and does not need to affect the thinking of the user. All that must be considered is that this particular surface, like the sphere, is negative on the inside and positive on the outside.

In the cross-section views of specific surfaces given later, we adopt the convention that the negative side of the surface is represented by the shaded portion of the illustrations.

Finite Limits to Surfaces

Many basic surfaces that can be defined are infinite in length. Normally, a user must specify two bounding planes and then must use all three surface designations when referring to a finite piece of this surface. To simplify this procedure, COG allows you to specify many surfaces as pseudosurfaces that include both the basic surface and the two bounding planes.

If surface number 17 is a right circular cylinder of radius 7.51, you would express the input definition as:

```
17 CYLINDER  7.51
```

This, by definition, is a cylinder with its axis along the x-axis and is infinite in extent. If you wanted a finite cylinder with its ends at x = 5.3 and x = 12.8, you would add this additional information thusly:

```
17 CYLINDER  7.51  5.3  12.8
```

This surface is negative at points within the finite cylinder and positive at all points outside the finite cylinder.

Surfaces with which this option may be used are indicated in the surface descriptions by including:

$\{x_{b1} \quad x_{b2}\}$

in the input specification where x_{b1} and x_{b2} are the two end points. The planes for the ends are drawn through these points and are perpendicular to the x-axis. The order that the two x values are given is not important—the surface exists at all points between the two.

Translation and Rotation Specifications

Each surface definition is given in its simplest form. Thus the cylinder, as illustrated above, is defined only by its radius. Its axis is taken to be on the x-axis. Now, not all cylinders that need to be used in a problem will be directed in this way. It is, therefore, necessary to provide additional data as to the location and orientation of the surface. This is done by providing a translation/rotation (TR) instruction to each surface description. The form of this specification is the same for each surface type. To understand this we need to know:

- Each surface is defined in its own coordinate system—(x', y', z')—independent of all other surfaces and the system coordinates. It is along the x'-axis, then, that the cylinder has its axis.

Defining the Geometry

- The specified problem has its own coordinates that are used throughout the entire description of the problem. This is the (x, y, z) coordinate system.

- The relationship between these two coordinate systems is determined by giving:

  — First, the location of the origin of the (x', y', z') system in terms of the (x, y, z) values at this origin. This is the point (x_0, y_0, z_0). This is the translation part of the specification.

  — Second, any point on the positive x'-axis, other than the origin, is given in terms of its (x, y, z) value. This is the point (x_1, y_1, z_1) and determines the first part of the rotation specification.

  — Third, any point on the positive y'-axis, other than the origin, is given in terms of its (x, y, z) value. This is the point (x_2, y_2, z_2) and completes the rotation specification. Because both systems are right-handed coordinate systems, it is unnecessary to specify a point on the z'-axis.

For surfaces in which the (x', y', z') and (x, y, z) systems are coincident, all the TR specifications may be omitted. For surfaces with the two systems translated from each other but with their axes parallel to each other, only the (x_0, y_0, z_0) values need be given. If the symmetry of the surface is such that an axis can be pointed in any direction, the necessary defining relation need not be specified. Thus, a sphere need only specify (x_0, y_0, z_0) and a cylinder need only specify (x_0, y_0, z_0) and (x_1, y_1, z_1).

The inputs added to a surface definition will then be:

(TR x_0 y_0 z_0) (x_1 y_1 z_1) (x_2 y_2 z_2)

In the following definitions of surface inputs, we take the liberty of condensing this to:

```
(TR ....)
```

As an example, take the finite right circular cylinder that was used in the previous section. If one point on the cylinder axis

occurs at (1.54, 3.29, 7.89)—and this is the point from which the two boundaries of the cylinder were measured—and if another point on the positive axis is at (104.56, 246.45, 534.23), then the surface could be specified as follows:

```
17 CYLINDER 7.51 5.3 12.8 TR 1.54 3.29 7.89 104.56 246.45 534.23
```

Planes and Figures Made up of Planes

Plane

A plane can be specified by any one of the following three methods. The first method is:

number [PLANE / PLA / P] [*x′,y′,z′*]$_1$ [*x′,y′,z′*]$_2$ (TR ....)

where the first point is located on the plane and the second point is located on the normal through the first point and at a point on the positive side of the plane.

The second method is:

number [PLANE / PLA / P] [*x′,y′,z′*]$_1$ [*x′,y′,z′*]$_2$ [*x′,y′,z′*]$_3$ [*x′,y′,z′*]$_4$ (TR ....)

where the points one, two, and three are all on the plane, but not in a straight line; and point four is anywhere on the positive side of the plane.

Defining the Geometry

The third method is:

$$number \quad \begin{bmatrix} \texttt{PLANE} \\ \texttt{PLA} \\ \texttt{P} \end{bmatrix} \quad c_0 \quad c_1 \quad c_2 \quad c_3 \quad (\texttt{TR} \ldots.)$$

where the c values are the coefficients in the analytic representation of the plane:

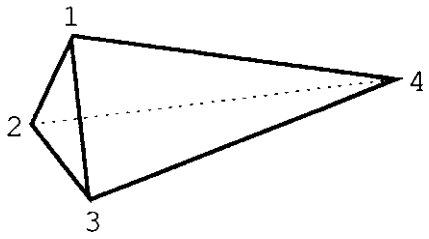
$$c_0 + c_1 x' + c_2 y' + c_3 z' = 0$$

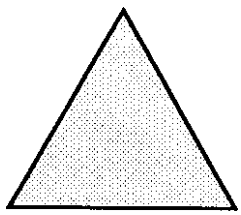
Tetrahedron

A tetrahedron, a four-sided polyhedron, can be specified by:

$$number \quad \begin{bmatrix} \texttt{TETRAHEDRON} \\ \texttt{TET} \\ \texttt{P4} \end{bmatrix} \quad [\ x', y', z'\ ]_1 \quad [\ x', y', z'\ ]_2 \quad [\ x', y', z'\ ]_3$$
$$[\ x', y', z'\ ]_4 \quad \{\ x'_{b1} \ x'_{b2}\ \} \quad (\texttt{TR} \ldots.)$$

with the four points falling at the corners of the figure.



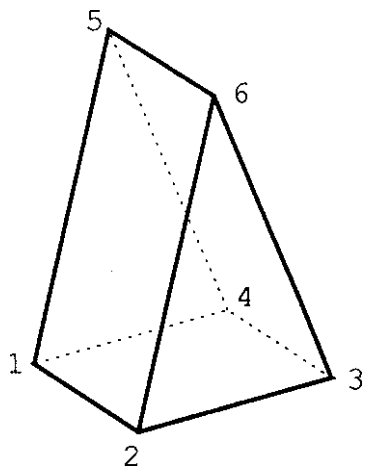


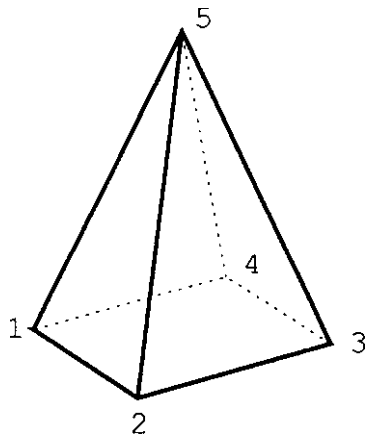
Pentahedron

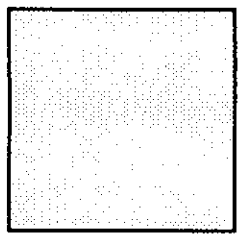
A pentahedron, a five-sided polyhedron, can be specified by:

number [PENTAHEDRON / PEN / P5] [x′,y′,z′]$_1$ .... [x′,y′,z′]$_5$
{ x′,y′,z′ }$_6$ { x'_{b1} x'_{b2} } (TR ....)

where the points are at the corners of the figure. If there are five corners, the figure is a pyramid. If the sixth point is included, the figure is a wedge. Note that the order of specifying the points should follow that indicated on the drawing below.







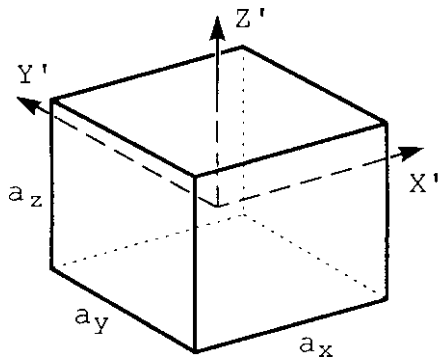
Defining the Geometry

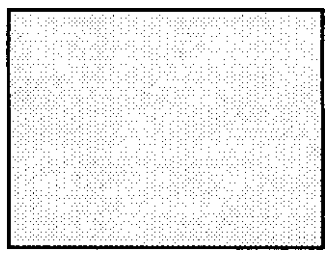
Box

A box is a special case of a six-sided polyhedron and is specified by:

number [BOX / B] a_x a_y a_z (TR ....)

with a_x, a_y, and a_z equal to the length of each side. The center of the box is at the origin and the sides are parallel to the coordinate planes.





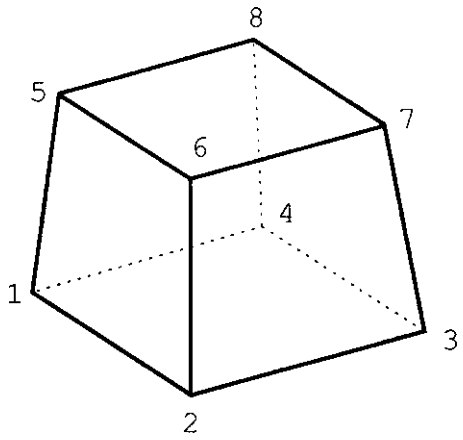
Hexahedron

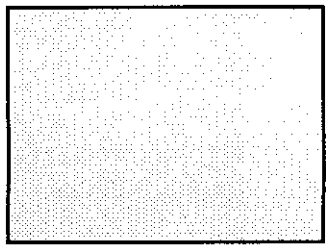
A hexahedron is a six-sided polyhedron specified by:

number [HEXAHEDRON / HEX / P6] [x', y', z']$_1$ .... [x', y', z']$_8$

{ x'_{b1} x'_{b2} } (TR ....)

The given points represent the eight corners of the figure and should be specified in the order indicated on the figure below.





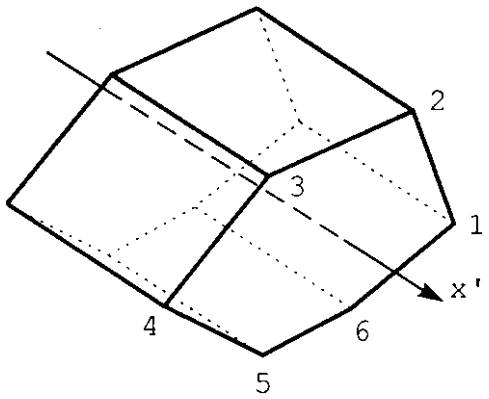
Defining the Geometry

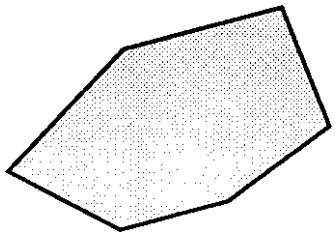
Right Prism

A general right prism with its axis parallel to the x'-axis is specified by:

$$number \begin{bmatrix} \texttt{PRISM} \\ \texttt{PRI} \end{bmatrix} \; number\text{-}points \; [\, y', z' \,]_1 \; [\, y', z' \,]_2 \; \ldots . \; [\, y', z' \,]_n \; \{ x'_{b1} \; x'_{b2} \} \; (\texttt{TR} \; \ldots .)$$

A right prism is defined by giving points in the (y', z') plane corresponding to the corners of the prism. The number of such edges is given along with the location of each corner. Note that the corner points should be given in the order as shown in the figure below. The defined prism must be non-reentrant—i.e., any straight line drawn so it starts within the figure and passes out through a side cannot again enter the figure.





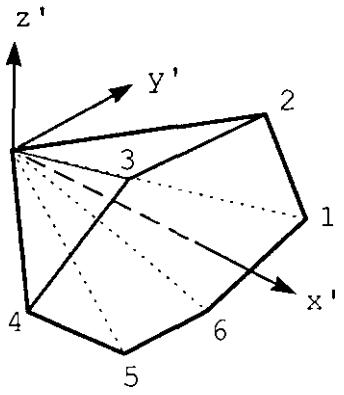
Right Pyramid

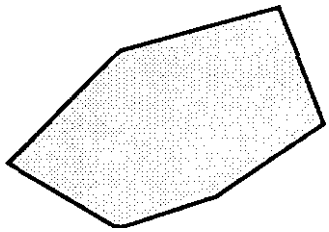
A general right pyramid with its apex at x' = 0 and its axis along the x'-axis is specified by:

number [PYRAMID / PYR] x'_{cx} *number-points* [y', z']$_1$ [y', z'

.... [y', z']$_n$ { x'_{b1} x'_{b2} } (TR ....)

A general right pyramid is defined by giving points in the (y', z') plane at $x'=x'_{cx}$ corresponding to the corners of the pyramid. The number of such edges is given along with the location of each corner. Note that the order of specifying the points should be as indicated in the figure below. As the prism, this figure must also be non-reentrant.





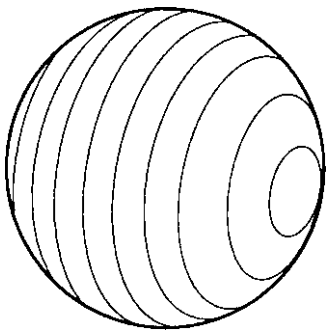
Defining the Geometry

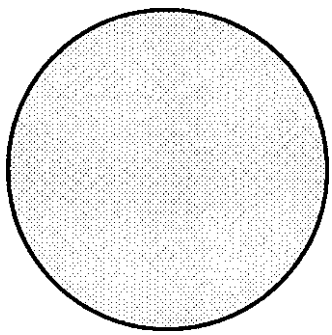
The Whole Family of Second-Order Surfaces

Sphere

A sphere is specified by:.

number [SPHERE / SPH / S] *radius* { x'_{b1} x'_{b2} } (TR ....)



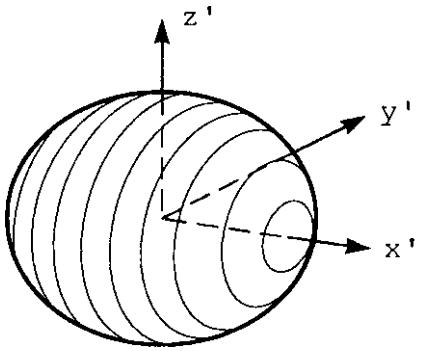


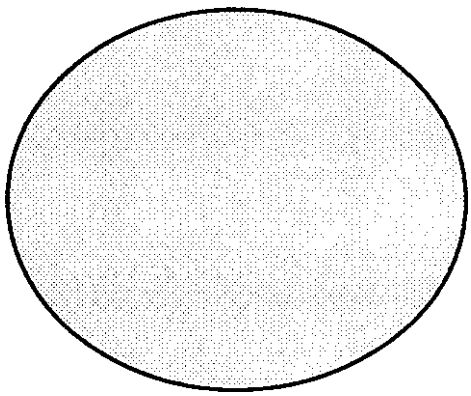
Ellipsoid

An ellipsoid can be specified by:

number [ELLIPSOID / ELL / E] *a b c* { x'_{b1} x'_{b2} } (TR ....)

The intercepts on the x'-axis are $\pm a$; on the y'-axis they are $\pm b$; and on the z'-axis they are $\pm c$.





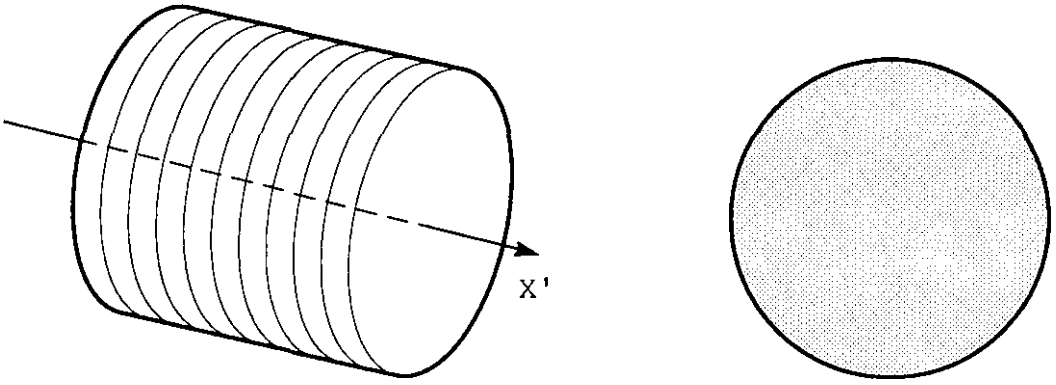
Defining the Geometry

Right Circular Cylinder

A right circular cylinder can be specified by:

number [CYLINDER / CYL / C] *radius* { x'_{b1} x'_{b2} } (TR ....)

The axis is coincident with the x'-axis.

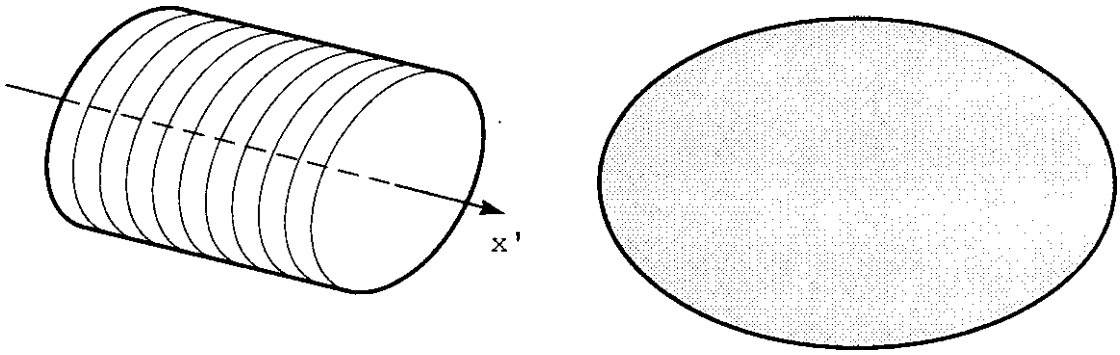


Right Elliptical Cylinder

A right elliptical cylinder can be specified by:

number [ECYLINDER / ECYL / EC] *b c* { x'_{b1} x'_{b2} } (TR ....)

The axis is coincident with the x'-axis. Intercepts along the y'-axis are at $\pm b$ and along the z'-axis are at $\pm c$.



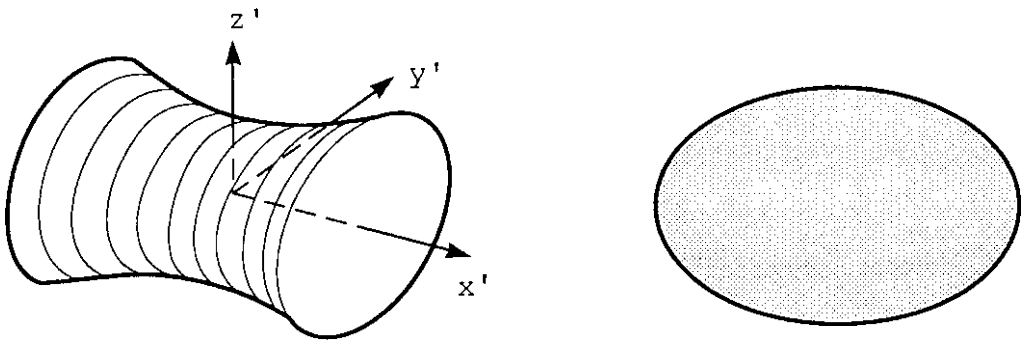
Defining the Geometry

Hyperboloid of One Sheet

A hyperboloid of one sheet can be specified by:

$$number \begin{bmatrix} \texttt{HYP1} \\ \texttt{H1} \end{bmatrix} \; a \; b \; c \; \{ x'_{b1} \; x'_{b2} \} \; (\texttt{TR} \; \ldots.)$$

The axis is coincident with the x'-axis and centered so that the minimum cross-sectional area occurs at $x' = 0$. Planes perpendicular to the x'-axis intersect the figure to form ellipses with semi-axes along the y' and z' directions. At $x' = 0$, the length of the semi-axes are equal to b in the y' direction and to c in the z' direction. At other values of x', the ellipses are similar in shape with the axes enlarged by a factor of $(1 + x^2/a^2)^{1/2}$.

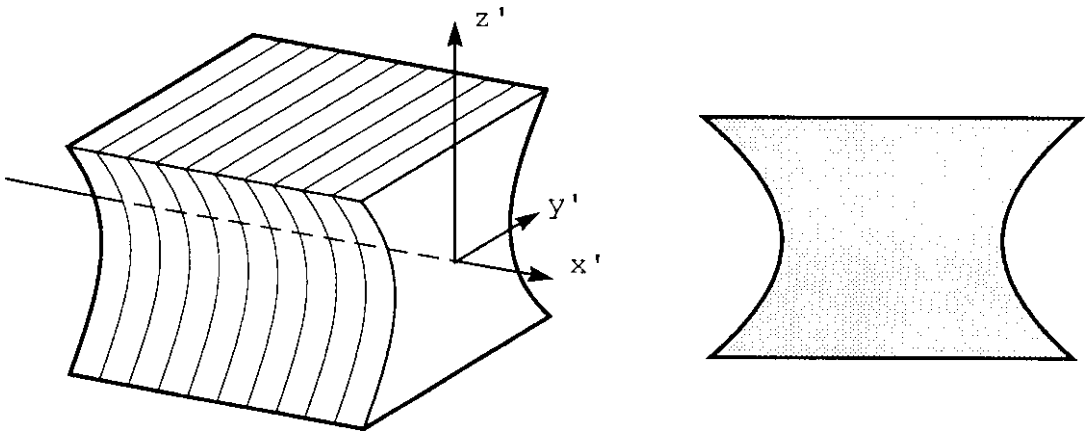


Hyperbolic Cylinder

A hyperbolic cylinder can be specified by:

number [HCYLINDER / HCYL / HC] *b c* { x'_{b1} x'_{b2} } (TR ....)

The hyperbolic cylinder is a hyperbola in the (y',z') plane extended along x'-axis without change. At $z' = 0$, the closest approach on the y'-axis is equal to b. For other values of z', this distance increases by a factor of $(1 + z'^2/c^2)^{1/2}$.

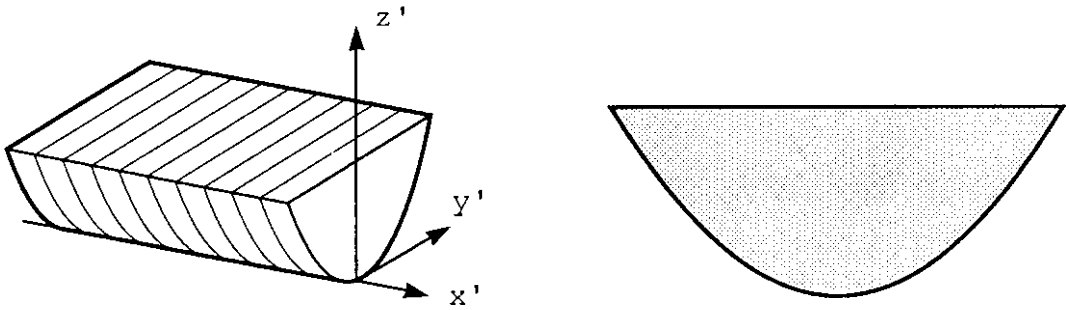


Defining the Geometry

Parabolic Cylinder

number [PCYLINDER / PCYL / PC] *b c* { x'_{b1} x'_{b2} } (TR ....)

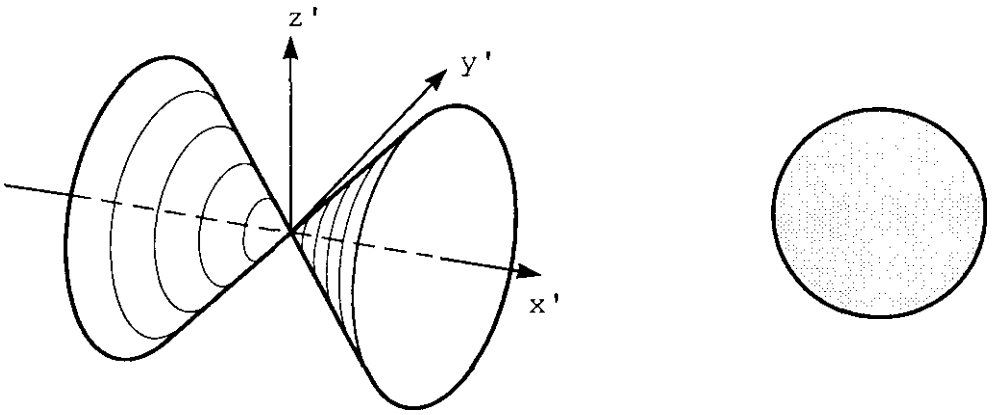
The parabolic cylinder is a parabola defined in the (y',z') plane and extended infinitely in the x' direction. At $y' = b, z' = 1/c$.



Right Circular Cone

number [CONE / CON / K] *a radius* { x'_{b1} x'_{b2} } (TR ....)

The axis of the right circular cone is coincident with the x'-axis, with the apex at the origin. The given radius is at x' = *a*.

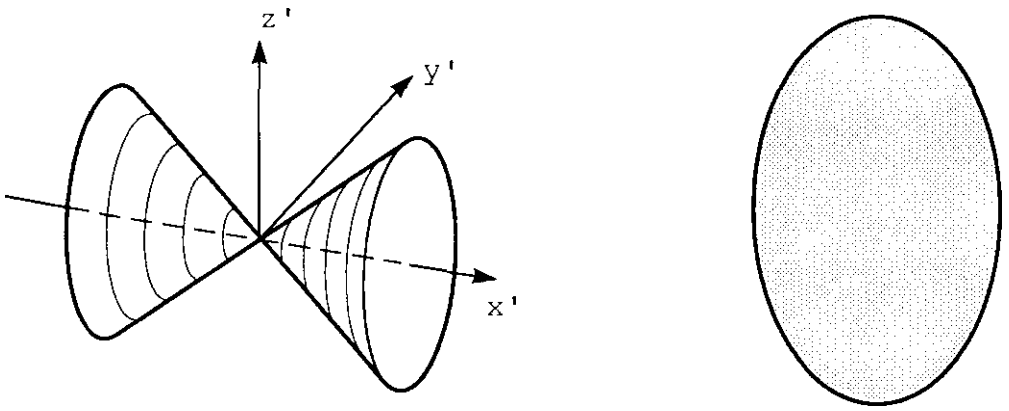


Defining the Geometry

Elliptical Cone

number [ECONE / ECON / EK] *a b c* { x'_{b1} x'_{b2} } (TR ....)

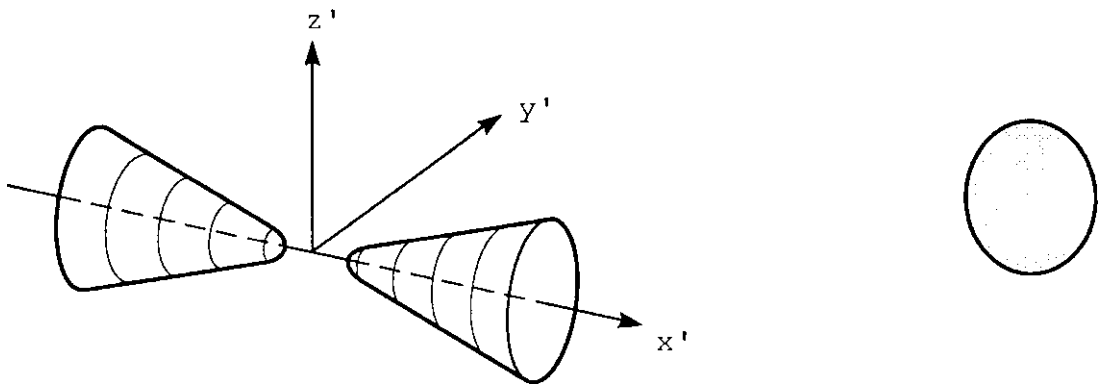
The axis of the elliptical cone is coincident with the x'-axis, with vertex at the origin. The cross section is an ellipse defined by *b*, the semi-axis in the y' direction, and *c*, the semi-axis in the z' direction—both measured at a distance *a* along the x'-axis from the vertex.



Hyperboloid of Two Sheets

$$number \begin{bmatrix} \text{HYP2} \\ \text{H2} \end{bmatrix} a\ b\ c\ \{ x'_{b1}\ x'_{b2} \}\ (\text{TR}\ \ldots.)$$

The axes of rotation of a hyperboloid of two sheets are coincident with the x'-axis and with the origin at the point of symmetry. The closest approach of the figure to the origin is at $\pm a$. A plane perpendicular to the x'-axis at x' = $a\sqrt{2}$ produces an ellipse with the semi-axis along the y'-axis equal to b and along the z'-axis equal to c.

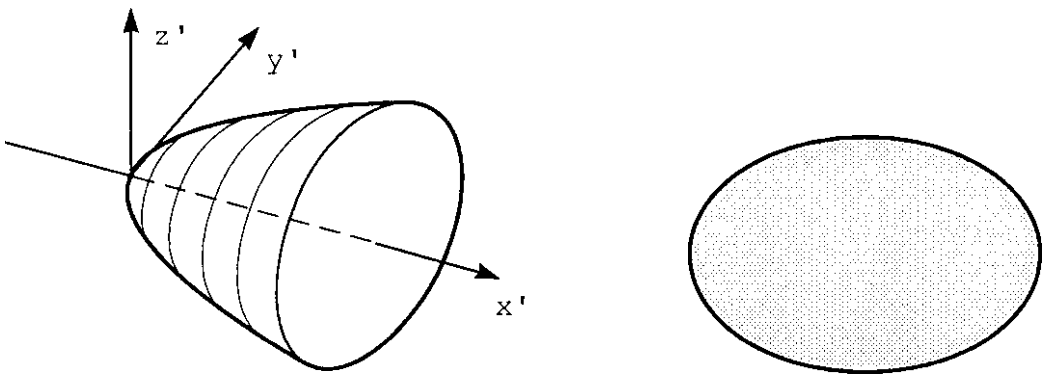


Defining the Geometry

Elliptic Paraboloid

number [EPAR / EP] *a b c* { x'_{b1} x'_{b2} } (TR ....)

The axis of rotation of the elliptic paraboloid is coincident with the x'-axis, with its vertex at the origin. At a distance of $1/a$ along the +x'-axis from the origin, the cross section of the figure defines a parabola with one semi-axis parallel to the y'-axis and equal to b and the other semi-axis parallel to the z'-axis and equal to c.

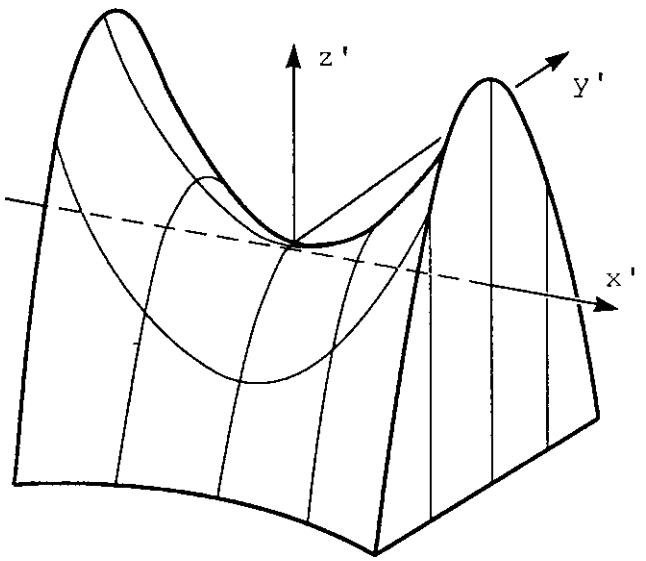


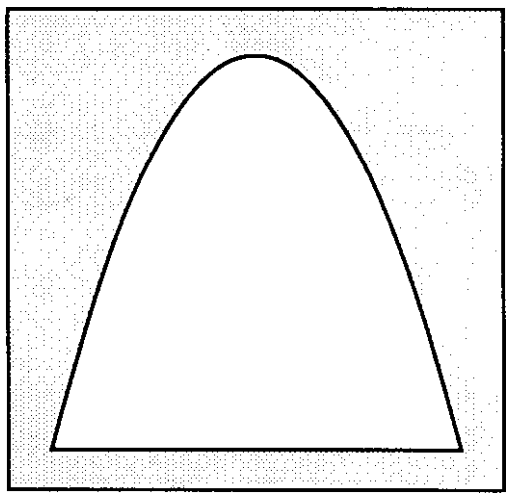
Hyperbolic Paraboloid

number [HPARABOLA / HPAR / HP] *a b c* (TR ....)

The axes of symmetry of the hyperbolic paraboloid are centered at the origin and coincident with the coordinate axes. The figure below is described by the equation:

$$(x'/a)^2 - (y'/b)^2 - cz' = 0$$





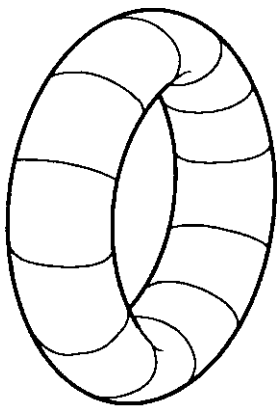
Defining the Geometry

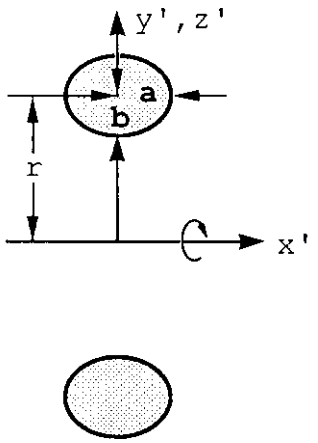
Special Figures of Rotation

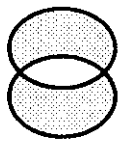
Torus

number [TORUS / TOR / T] *r* *a* {*b*} (TR ....)

The elliptical torus is formed by the rotation around the x'-axis of an ellipse defined in the (x',y') plane. This ellipse has its center on the y'-axis a distance *r* from the origin, a semi-axis parallel to the x'-axis equal to *a*, and a semi-axis along the y'-axis equal to *b*. If *b* is omitted in the input, *b* is set equal to *a* and a circular torus is defined. Note that the degenerate form (when *b*>*r*) excludes the central region.







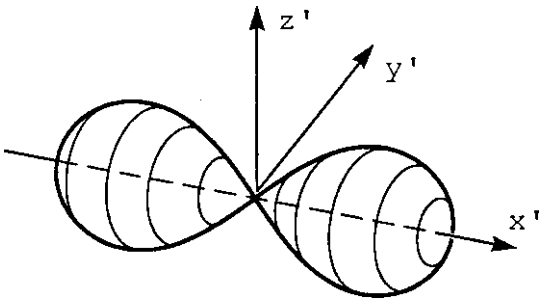
Degenerate

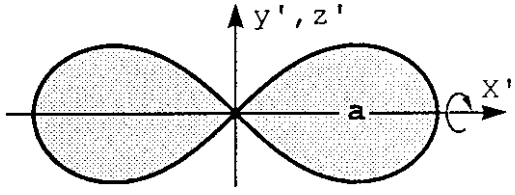
Lemniscate Rotated About the x'-axis

number LEM-X *a* (TR ....)

The Lemniscate of Bernoulli is rotated about the x'-axis. The constant *a* measures the intercept with the figure on the x'-axis. The plane curve has the equation:

$$((x')^2 + (y')^2)^2 = a^2((x')^2 - (y')^2)$$





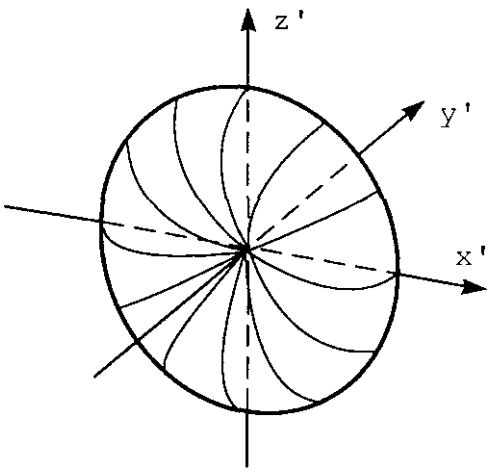
Defining the Geometry

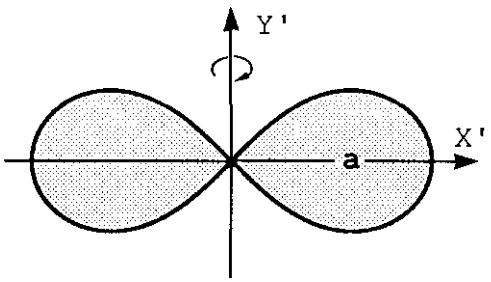
Lemniscate Rotated About the y'-axis

number LEM-Y *a* (TR ....)

The Lemniscate of Bernoulli is rotated about the y'-axis. The constant *a* measures the intercept with the figure on the x'-axis. The plane curve has the equation:

$$((x')^2 + (y')^2)^2 = a^2((x')^2 - (y')^2)$$





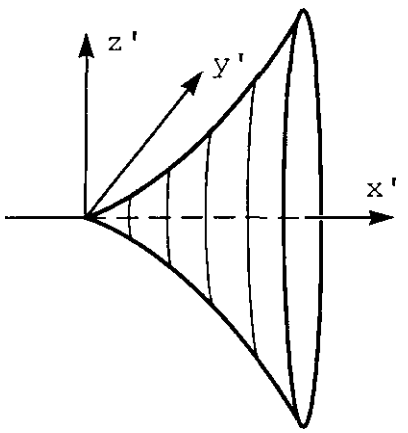
Semicubical Parabola Rotated about the x'-axis

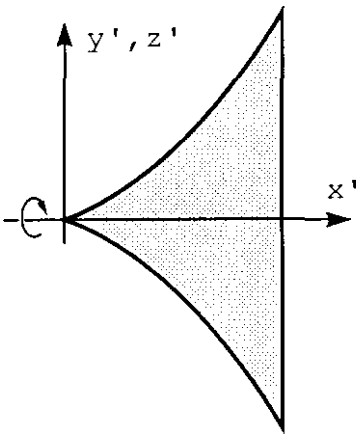
A semicubical parabola is rotated about the x'-axis and specified by:

number [SPARABOLA / SPAR] *a* (TR ....)

The equation of the plane curve is:

$$(y')^2 = a\ (x')^{1/3}$$





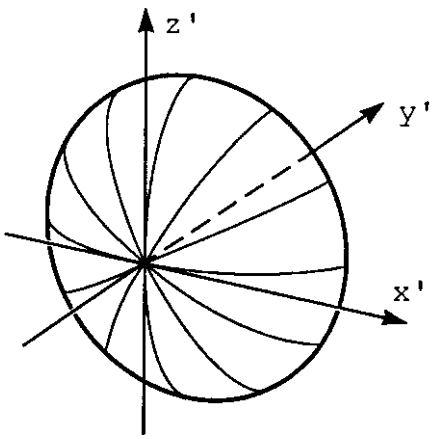
Defining the Geometry

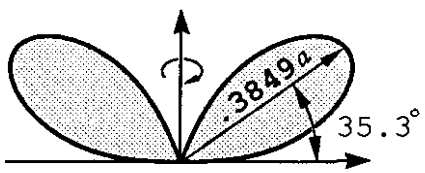
Bifolium Rotated About the y'-axis

number [BIFOLIUM / BIF] *a* (TR ....)

The equation of the rotated plane curve for a bifolium rotated about the y'-axis is:

$$((x')^2 + (y')^2)^2 = a(x')^2(y')$$



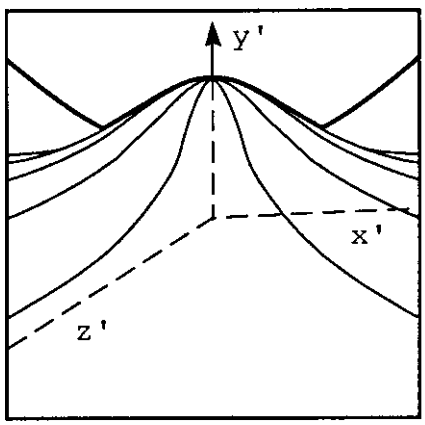


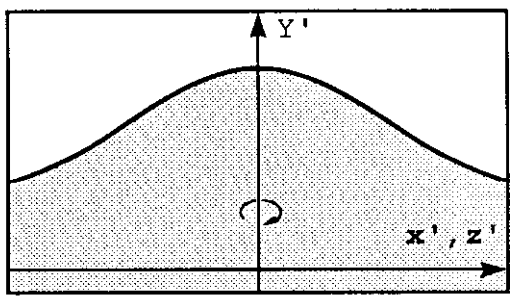
Witch Rotated About the y'-axis

number [WITCH / WIT] *a* (TR ....)

The plane curve known as the Witch of Agnesi is rotated about the y'-axis. The equation of the plane curve is:

$$y' = 8a^3 / ((x')^2 + 4a^2)$$





Note coordinate directions

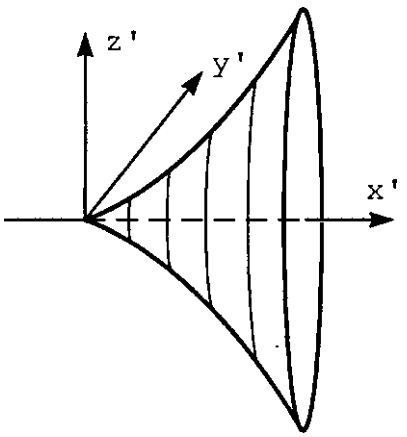
Defining the Geometry

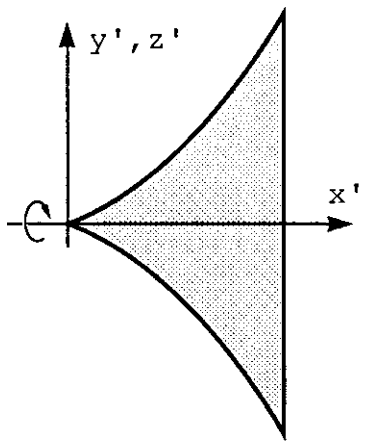
Cissoid Rotated About the x'-axis

number [CISSOID / CIS] *a* (TR ....)

The plane figure know as the Cissoid of Diocles is rotated about the x'-axis. The equation of the plane figure is:

$$(y')^2 = (x')^3 / (a - x')$$



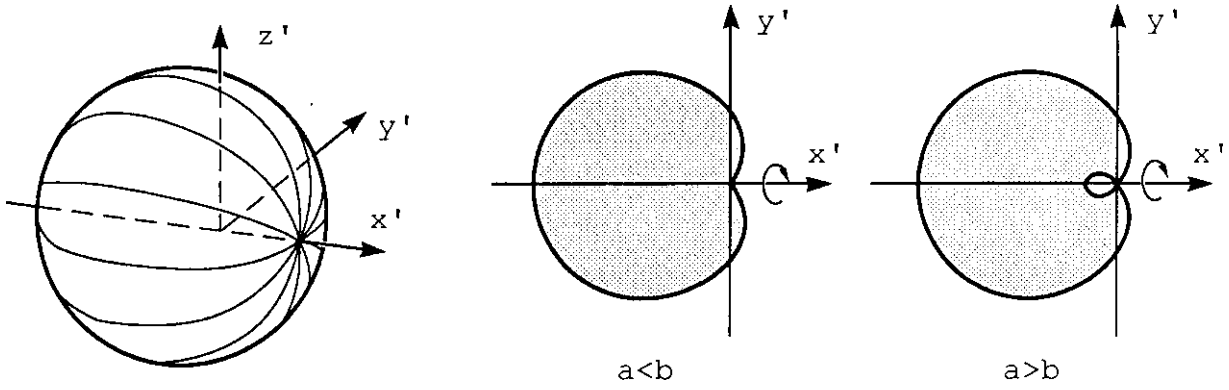


Limacon Rotated About the x'-axis

number [LIMACON / LIM] *a b* (TR ....)

The plane curve known as the Limacon of Pascal is rotated about the x'-axis. Intercepts on the x'-axis are at $-a+b$, $-a-b$, and, if a is greater than b, at zero. Intercepts on the y'-axis are at $+b$ and $-b$. Function is negative within the figure except for the case where a is greater than b, which yields an interior region (similar to the degenerate torus) that is evaluated to be positive. The equation of the plane figure is:

$$((x')^2 + (y')^2 + ax')^2 = b^2 ((x')^2 + (y')^2)$$



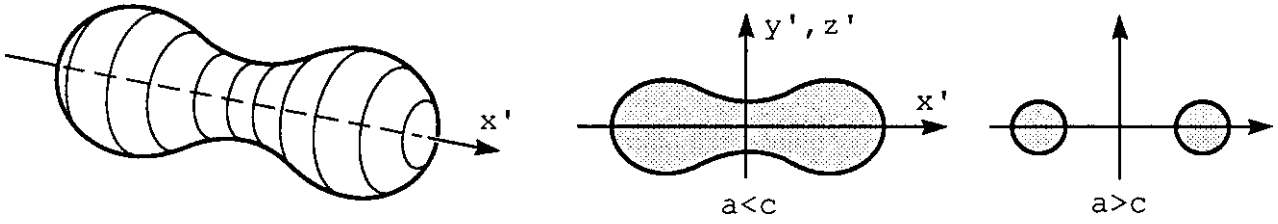
Defining the Geometry

Ovals Rotated About the x'-axis

number [OVALS / OVAL / OVA] *a c* (TR ....)

The plane curve known as the Ovals of Cassini is rotated about the x'-axis. Foci are at $(-a,0)$ and $(a,0)$. The product of the distances from any point on the curve to each of the foci is equal to c^2. The total length of the figure along the x'-axis is equal to $2(a^2 + c^2)^{1/2}$. Along the y'-axis, the length is equal to $2(c^2 - a^2)^{1/2}$. For a less than c, the figure looks like a dumb-bell with the inside determined to be negative. For a equal c, the figure becomes a Lemniscate rotated about the x'-axis. For a greater than c, the figure resolves itself into two egg-shaped surfaces. The equation of the plane figure is:

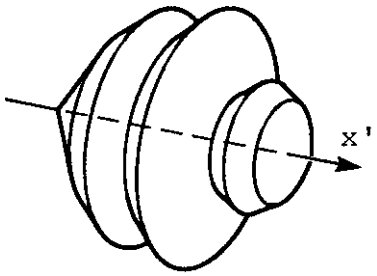
$$((x')^2 + (y')^2 + a^2)^2 - 4\ a^2\ (x')^2 = c^4$$

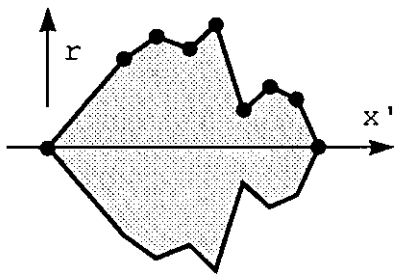


Curve Made of Straight-Line Segments Rotated About the x'-axis

number [REVOLUTION / REV / R] *number-points* [*x',r'*]$_1$ [*x',r'*]$_2$ [*x',r'*]$_3$ ... [*x',r'*]$_n$ (TR ....)

A series of straight lines are used to approximate a curve. These are then rotated about the x'-axis to form a solid figure. The *x'* values of the points must be given in increasing order. If the first point does not have a zero radius, or if the distance between the first and second point is less than 0.005, the figure is bounded by a plane at the first value of *x'*. A similar termination is also done for the last given value of *x'*.





Defining the Geometry

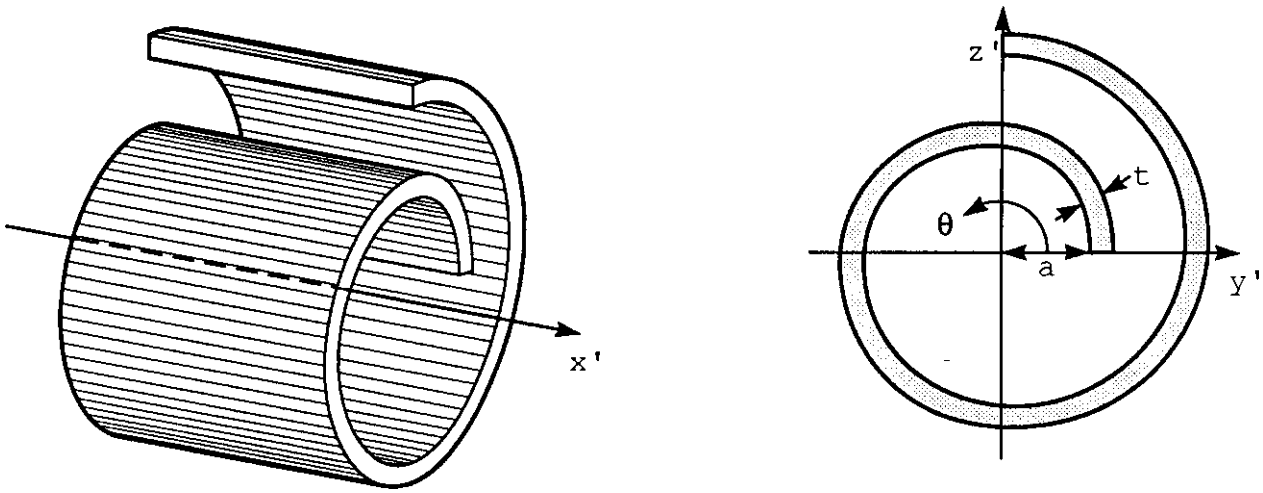
Special Surfaces

Spiral-Wound Foil

number [SPIRAL / SPI] *r a t* θ_1 θ_2 x'_{b1} x'_{b2} (TR ....)

The above definition is a simulation of a flat foil wound into a spiral with the axis of rotation coincident with the x'-axis. Looking from a point on the positive x'-axis toward the origin, the rotation is counter-clockwise.

The foil is defined by *r*, the distance from the origin to the inside of the first turn of the foil along the +y'-axis; *a*, the distance from the previous point to the same position on the foil at the start of the second turn; *t*, the thickness of the foil; θ_1 the angle defining the start of the foil (zero if the first start is on the +y'-axis as described); θ_2 the angle defining the end of the foil; and the *x'* positions at the start and end of the foil (note that these are not optional). The foil is simulated by a series of prisms each covering a five-degree change in θ.

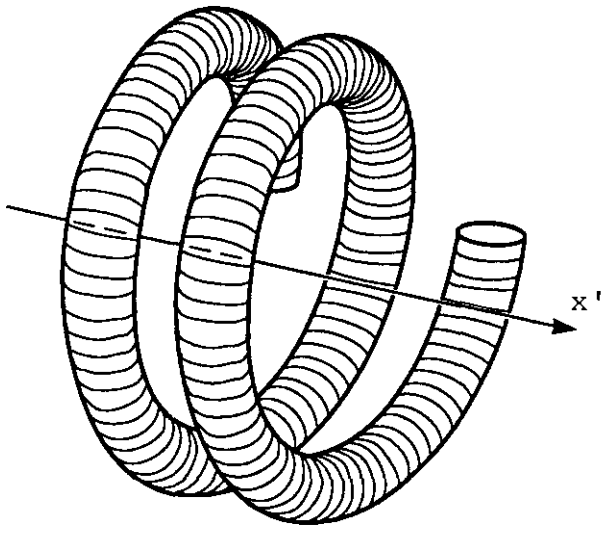


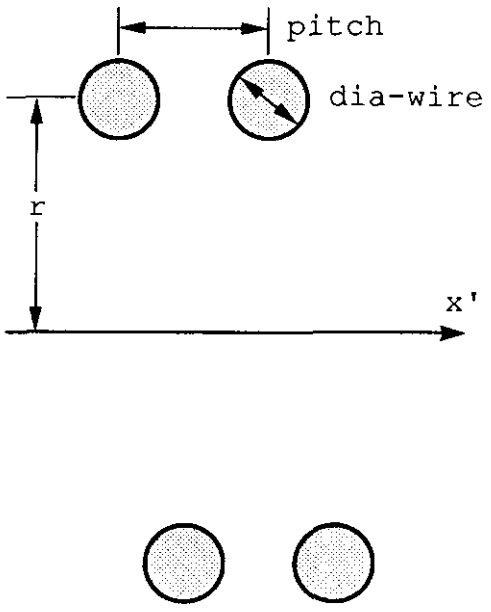
Helix with a Circular Cross Section

number [HELIX / HEL] *r pitch diameter-wire* θ_1 θ_2
x'_{b1} x'_{b2} (TR ....)

The above definition is a simulation of a circular cross-section wire wound on the outside of a right circular cylinder, whose axis is coincident with the x'-axis. At a point on the plus x'-axis looking backward toward the origin, the helix is wound in a counter-clockwise direction.

The helix is defined by its radius *r* (the distance from the x'-axis to the center of the wire), its *pitch* (the distance parallel to the x'-axis needed to make one revolution of the wire), the *diameter* of the wire, the starting value of θ (θ is zero at the starting point on the +y'-axis at x' = 0 and increases along the +x'-axis), and the ending value of θ. The surface is simulated by a series of short right-circular cylinders, each running over approximately five degrees in θ.





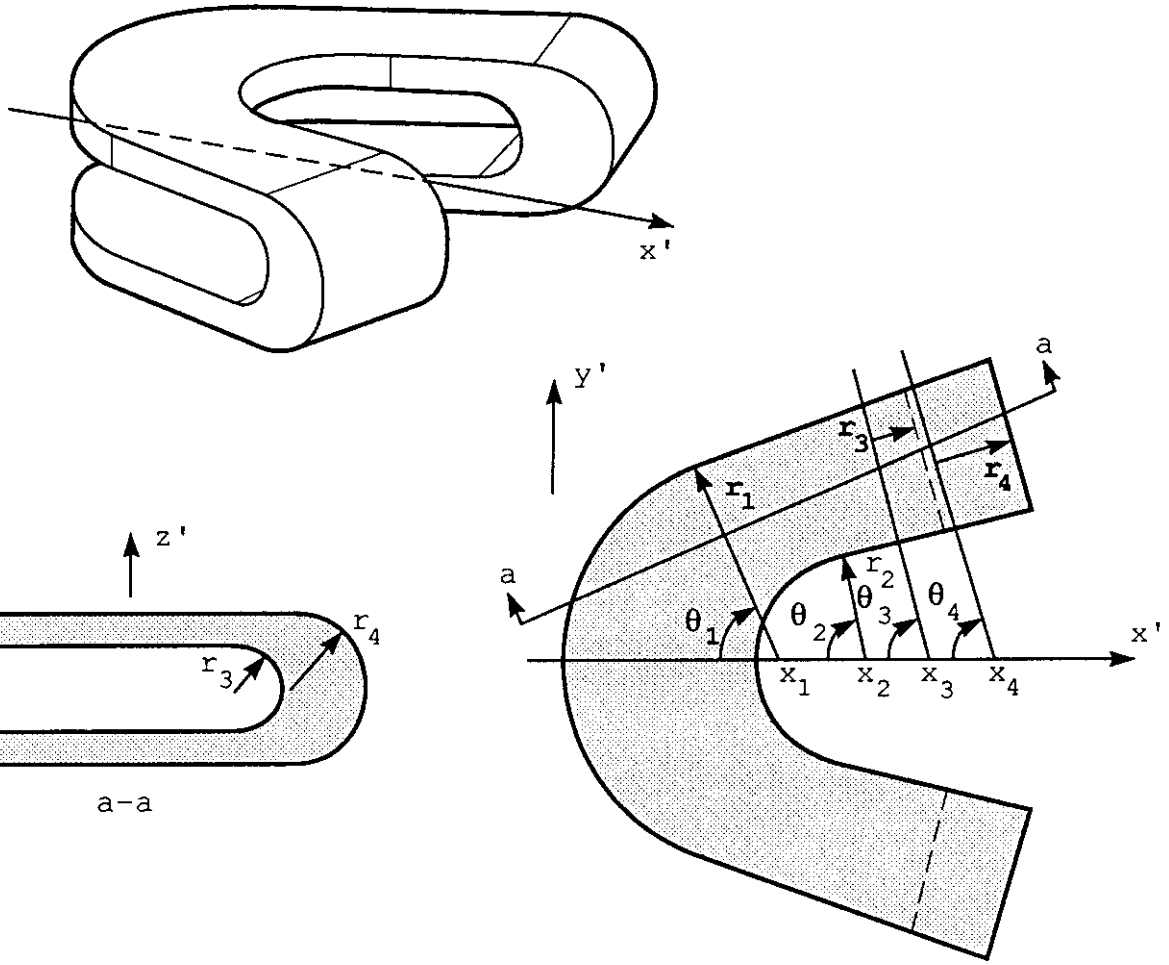
Defining the Geometry

Yin-Yang Coil

```
number [YIN-YANG]   [ x', r', θ' ]1   [ x', r', θ' ]2
       [YIN     ]   [ x', r', θ' ]3   [ x', r', θ' ]4
                    (TR ....)
```

The above definition is a simulation of half a Yin-Yang coil. The coil has its center line run along the x'-axis with the planes defining the top and bottom of the coil parallel to the (x',y') plane.

The figure below illustrates the defining quantities of the coil. The points 1 through 4 do not necessarily have to be in the order shown.



Topographical Surface

number [TOPOGRAPHICAL / TOP]

$X = x_1 \quad x_2 \quad x_3 \quad x_4 \quad \ldots\ldots$

$Y = y_1 \quad y_2 \quad y_3 \quad y_4 \quad \ldots\ldots$

$Z = z\ (x_1, y_1) \quad z\ (x_2, y_1)$

$z\ (x_3, y_1) \quad \ldots\ldots\ldots\ldots$

$z\ (x_n, y_1) \quad z\ (x_1, y_2)$

$z\ (x_2, y_2) \quad \ldots\ldots\ldots\ldots$

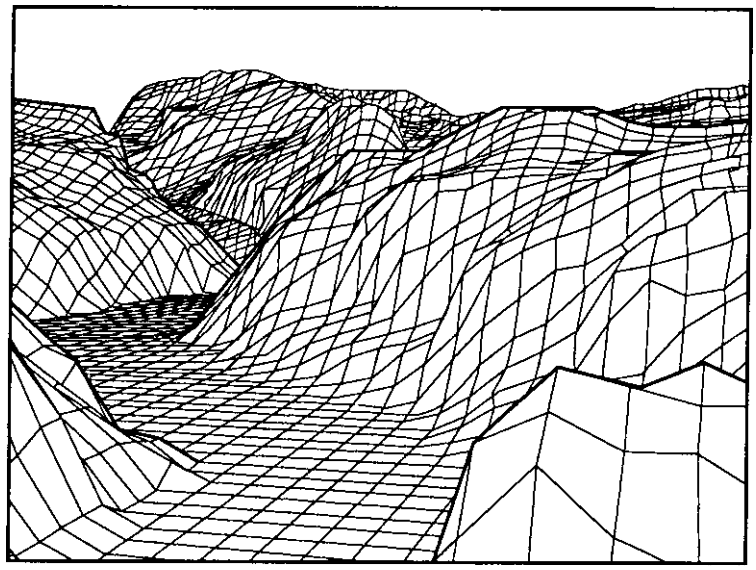
$\ldots\ldots\ldots\ldots \quad z\ (x_n, y_m)$

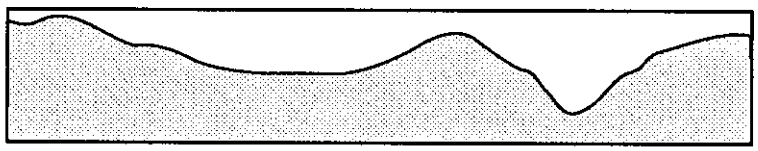
The input format above provides an approximation of a topographical surface by specifying an x-grid (E-W direction) and a y-grid (N-S direction) and then specifying the z-value (altitude) at every intersection point of the x- and y-grid. The surface fit in each section of this grid assumes that the curve is linear in x and, independently, linear in y. This yields a second-order surface valid within each section of the defined grid.

Note: No rotation or translation is allowed with this figure.

The United States Geological survey data are available for all of the continental United States and can be converted into the above COG input format.

Defining the Geometry





General Analytic Surfaces

A user may specify that a given surface is defined by an input analytic equation containing terms up to the fourth degree.

number [ANALYTIC / ANA / A] [*coeff*, TERM]$_1$ {*coeff*, TERM}$_2$
{*coeff*, TERM}$_3$ .... (TR ....)

In the format input above, the *coeff* (coefficient) input represents a numerical value associated with a given algebraic term that is represented in the input by a BCD word TERM. There are as many input pairs used as needed to specify the given surface. The BCD word takes on forms like X, XX, XZ, and ZZXY to indicate x, x^2, xz, and xyz^2. The code does not care about how the letters are ordered in this BCD word—it will recognize that XYZ, XZY, YZX, YXZ, ZXY, and ZYX all represent the xyz term.

Constant terms are recognized by the word CONSTANT, or CONST, or, in cases where no misunderstandings could result, just by omitting the BCD word altogether. A numerical value of 1.0 may also be omitted if this would not cause a misunderstanding in the code inputs. The same term may be input more than once—in this case the code will sum the input coefficients for all the inputs of this term.

Example. The sphere of radius 5 located at (–5,4,0) is given by the following surface equation:

$$(x - 5)^2 + (y + 4)^2 + z^2 - 5^2 = 0$$

This could be specified in the input, assuming that this is surface number 1023, by the following:

```
1023 ANALYTIC 1.0 XX  -10.0 X  25.0 CONST 1.0 YY 8.0 Y
             16.0 CONST 1.0 ZZ -25.0 CONST
```

Defining the Geometry

Direct Coupling to Internal Data Storage

number [DIRECT / DIR / D] *internal-type* [*data* ....]

The input type above allows data to be stored directly into the core memory without any transformation or interpretation. Needless to say, only a user well versed in the internal operations of the geometry module should attempt to use this option.

As an example, you need to define a geodesic dome made up of 100 surfaces. Internal surface type 19 is capable of describing a pseudosurface formed of *n* plane surfaces. This surface type is used by the module when handling tetrahedrons, pentahedrons, boxes, hexahedrons, right prisms, and right pyramids. You know that to define a dome all you need do is to specify *n* as 100 and then give the coefficients that describe 100 plane surfaces. Input of these values in the proper order accomplishes your objective without excessive geometry input descriptions or without rewriting the basic COG program.

It is beyond the scope of this manual to describe enough detail to allow users to make use of this option, but they may wish to remember that this option exists and to discuss with the COG programming group any problems they may encounter in the future that could profit from its use.

Sectors and Related Items

Sector Definitions

Regular Sector

The GEOMETRY block of input first contains information that defines individual sectors by using the surfaces that have been specified.

Basically, these inputs are of the form:

SECTOR *number* *name* $\pm s_1$ $\pm s_2$ $\pm s_3$ ....

where:

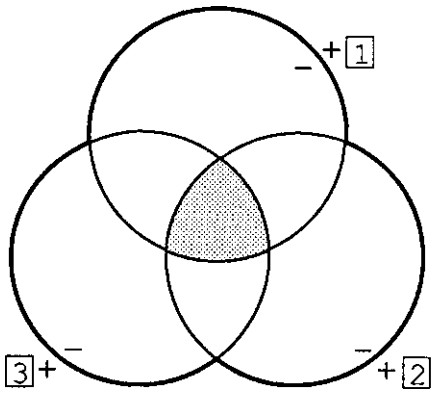
number is the sector number provided by the user. This is any integer greater than or equal to minus one, with the special stipulation that minus one signifies that the sector contains an *infinite absorber* and that zero signifies a void. An infinite absorber is a fictitious material with an infinitely large absorbing cross section. Any particle entering the cross-section region will be terminated, and any point flux estimation made through such a region will result in a zero contribution. If ASSIGN inputs are provided, the material number and statistical region number are taken from them to go with the sector number described above. If no ASSIGN statements goes with the used sector number, then both the material and region numbers are set equal to the sector number. For the case in which an infinite absorber is specified, the statistical region number will be set to zero.

name is an input name that the user wishes to be attached to the sector. It has no use in the code except to be printed out to help remind the user of what this specific sector represents. Remember that only the left-most eight characters will be saved and printed.

The remaining part of the specification is the definition of the sector itself. The quantities $\pm s_1$ $\pm s_2$ $\pm s_3$ .... represent surface numbers. $+s_1$ means that the sector is on the positive side of surface number s_1. $-s_2$ means that the sector is on the negative side of surface number s_2. As many surfaces, with the appropriate signs, as are needed to completely define the sector

Defining the Geometry

should be specified. In the example below, the sector that is within all three surfaces would be defined as:




```
SECTOR 7 INSIDE −1 −2 −3
```

No two sectors should overlap. Using the surfaces illustrated in the previous figure, the input

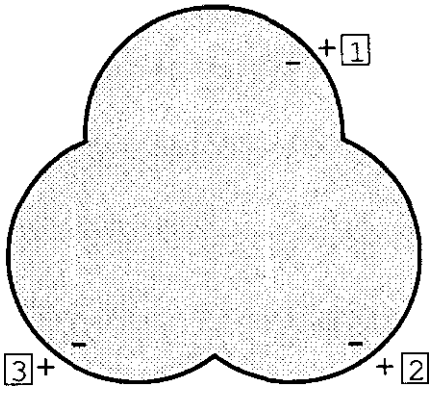
```
SECTOR 7 INSIDE −1 −2 −3
SECTOR 8 ERR7      −2
```

would produce an error message, since sector 8 and sector 7 have some of the same volume included in each.

In some cases it is convenient to combine several separate sector definitions into one sector definition. To accomplish this, the general sector definition is expanded to

```
SECTOR  number  name ±s1  ±s2 ...
(OR  ±s11  ±s12 ....)
(OR  ±s21  ±s22 ....) ....
```

These separate definitions may overlap without errors. As an example;




```
SECTOR 7 INSIDE -1 OR -2 OR -3
```

would include all the indicated volumes within sector 7 and would not produce an error.

Internal code operations, invisible to the user, re-order the sequence of surfaces that define a sector so that those most quickly calculated occur first. During code operation, this saves running time in the process of determining if a point is in a given sector. In the same manner, the code learns which sectors are the most probable ones to be entered when a particle leaves a given sector through a given surface. The search for a new sector starts with these. Therefore, in setting up a problem, no thought need be given to the ordering of sectors or to the ordering of surfaces within sectors.

Default Fill

For a complete geometry specification of a problem, all parts of physical space should be included in the set of sector definitions. For certain geometry descriptions, this becomes very difficult, since it takes a lot of definitions to catch all the little left-over pieces. The COG geometry package provides a sector that picks up all the volume that is not in one of the defined sectors. This is referred to as the fill sector. It will exist without user specification and will be assigned a default sector number of zero—thus, containing a void.

Defining the Geometry

The user may assign a different sector number to the fill by providing the following input:

```
FILL   number
```

Only one such statement is allowed, and the sector will then contain the material and region numbers assigned to go with this specific sector number.

Specifying Boundary Conditions

Vacuum

A boundary condition imposes a physical boundary to the problem being run. The type of the boundary will determine what will happen to a particle that hits it. A vacuum boundary implies that the particle will enter into a void and keep on going forever, never to return to the defined geometry again. Because this is the case, the random walk of the particle will be terminated. If you place a point estimator inside a vacuum boundary, COG will not calculate an attenuation for the path from the boundary to the detector. If you specify a source within such a boundary, the code will let the particles move as if they were in a void hoping they will enter the defined geometry.

COG specifies the boundary as the surface of a defined volume. The volume is defined in the same way as a sector. The appropriate input is:

BOUNDARY VACUUM $\pm s_1$ $\pm s_2$ ....(OR $\pm s_{11}$ $\pm s_{12}$ ...)...

If necessary, more than one boundary specification may be made.

As long as the fill material is left as a void, it should be evident that an automatic vacuum boundary condition exists at the outside of each defined problem. In most problems, the use of the true vacuum boundary condition can save calculation time.

Reflecting

A reflecting boundary condition produces specular reflection back into the problem with no weight or energy change. Specular reflection is defined so that the returned particle is at the same angle from the normal to the boundary as the incident particle was and so that the incident direction, the normal, and the reflected direction lie in the same plane.

In a physical sense, this boundary lies at a place where the same particle flux distribution exists on each side of the boundary. Thus, a problem with a symmetrical result around a surface need only calculate the random walk on one side of the surface.

The input for this boundary is:

BOUNDARY REFLECTING $\pm s_1$ $\pm s_2$ ....(OR $\pm s_{11}$ $\pm s_{12}$ ...)...

A source placed within a reflecting boundary will yield a fatal error message and terminate the problem.

If you use this boundary, COG imposed a restriction on problem solutions when a point flux estimator is used. This type of estimator *looks* at each collision site in the random walk and then makes an estimation of the contribution to the flux at the specified point. If there is a reflecting boundary in the problem, some of these collisions are not evaluated—those on the other side of the boundary. The point detector will indicate a lower result because of these missing collisions. If there were always just one boundary, and it were always a plane, it would be simple to have the code evaluate the point estimation correctly. However, it is possible to specify many such boundaries in one problem; and there are, consequently, many difficulties in making the code operate correctly under such cases. When a problem contains both reflecting boundaries and point estimators, an error warning is printed to remind the user of these facts. If the contribution from the missing volume would be small, the problem solution could be acceptable. Users must decide that condition for themselves.

Albedo

An albedo boundary is somewhat like a reflecting boundary except that the angle, energy, and weight of the reflected particle is a function of the incident energy, the direction, and the weight of that particle. The relationship between these are taken from a defined albedo that must be input to COG using the ALBEDOIN block of input data (see page 17). The format for this boundary is:

BOUNDARY *number* ALBEDO $\pm s_1$ $\pm s_2$ ....(OR $\pm s_{11}$ $\pm s_{12}$ ...)...

where *number* is an identifying number of the specific albedo to be used. More than one albedo may be used in a problem.

Defining the Geometry

If you use this option with point estimators, COG produces a fatal input error. Location of a source within the albedo boundary is also fatal.

This option is not used frequently but has been used in the past for several problem types with great success. A number of years ago, workers at Oak Ridge calculated a civil defense problem involving radiation transmission down a tunnel and around corners. Instead of evaluating the collisions in the tunnel walls for each incident particle, the walls were replaced with albedo surfaces. Their problems were then reduced to following a series of albedo scatterings. The results they obtained with the full Monte Carlo calculations in the walls were very close to those obtained using the albedo method. They achieved substantial reductions in computing time.[15]

In the same way, reflecting neutrons back into a critical assembly by an infinite water reflector may be represented by albedo boundaries with the assurance that the calculated system multiplication will always be larger than the exact physical multiplication.[16]

Others also achieved success in following reflected light transmitted down channels by using the albedo method.[17]

One problem with the albedo option is that the returned particle is assumed to come back into the system at the same point that the initial particle exited the system.

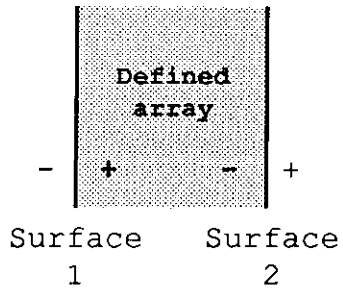
Periodic

The periodic boundary is specified by:

BOUNDARY *surface-number* PERIODIC $\pm s_1$ $\pm s_2$ ...(OR ...)

where *surface-number* refers to a surface defined in the SURFACES block of inputs.

This boundary is used occasionally in criticality problems when the multiplication of an infinite number of stacked arrays is desired. A simple one-dimensional problem would look like:



Periodic boundaries are defined on the left, on the negative side of surface one, and on the right, the positive side of surface two, thusly:

```
BOUNDARY 2  PERIODIC  -1
BOUNDARY 1  PERIODIC  +2
```

When a particle hits surface two, it is moved to a point on surface one where the normal or the struck boundary intercepts surface one. The direction, weight, and energy of the particle are unchanged and the particle is allowed to continue its trajectory. Likewise, when a particle hits surface one, it is moved to surface two.

The user may extend his example to four or six planes or to surfaces that are not planes. Fatal errors are encountered if a source is located within the boundary or if point estimators are used.

Making Complex Geometries Simple

The Unit Definition

With a knowledge of surfaces, sectors, the fill sector, and boundary conditions,you know enough to define any geometry to the code. For simple problems, you need use no additional techniques; but when the geometry starts to become very complex, problems start to occur. These are not problems of doing the job correctly, these become problems associated with the complexity of large amounts of data. Just imagine trying to keep track of some three or four hundred sectors. Not only do *you* have problems, the code itself becomes burdened with the evaluation of a multitude of surfaces.

Defining the Geometry

There is no general way to define such a problem with fewer sectors, but there is a way to take the problem in small bites. Thus, we lead into the concept of a UNIT.

We define UNIT as a piece of volume, just as SECTOR. In fact, units are defined right along with sectors. When a particle enters the unit, instead of finding a specified material it finds a specified set of SECTOR definitions. These are separate from the SECTOR definitions the particle was just traveling in. As long as the particle stays within the UNIT, it can forget all about the original set of SECTOR definitions and just keep track of the definitions within the UNIT. For those who write computer programs, it is just like dropping into a subroutine. For simple units, the required code input is:

```
UNIT  number  name  ±s1  ±s2  ±s3  .....
```

where *number* is a positive integer the user assigns to UNIT for identification purposes; *name* is just an identification name to be printed with the code output; and the surface definitions are the same as for a SECTOR.

Now this unit must be provided with a definition. This starts with the statement:

```
DEFINE  UNIT  number
```

after which the unit definition, in terms of sectors and fill statements, is provided. To show this a little better, the following illustrates the pattern for a case where two different units are employed:

```
SECTOR  1   AA1   .......
SECTOR  2   AA2   .......
SECTOR  3   AA3   .......
UNIT    4   AA4   .......
SECTOR  5   AA5   .......
SECTOR  6   AA6   .......
UNIT    7   AA7   .......
SECTOR  8   AA8   .......
FILL    9
BOUNDARY  VACUUM  .........

DEFINE  UNIT  4
  SECTOR  17  BB1   .......
  SECTOR  18  BB2   .......
  FILL    19
```

```
DEFINE  UNIT  7
  SECTOR  20  CC1  .......
  SECTOR  21  CC2  .......
  SECTOR  22  CC3  .......
  SECTOR  23  CC4  .......
```

When a particle is being tracked in the original definition, we refer to this as operating in level-0. When the particle moves into unit 4 or into unit 7, the particle begins moving in level-1.

In the definition of a unit, you can reference one or more other units . Thus, unit 4 may have been defined as:

```
DEFINE  UNIT  4
  SECTOR  17  BB1  .......
  SECTOR  18  BB2  .......
  UNIT    30  BB3  .......
  FILL    19
```

There would then have to be a definition for unit 30. Entrance into unit 30 would transfer the particle to level-2. It is possible to go to level-9 in the existing version of COG. The user does not really have to understand about levels except when the levels reach a point that they are greater than 9, and a fatal error to that effect is printed. When an error occurs in the geometric setup, the level number is also printed in the resulting fatal error statement.

Note: Boundary definitions can occur only at level-0.

Needless to say, if you use a UNIT statement calling for a given unit number, you must provide the definition of that unit.

A number of different unit statements may refer to the same UNIT definition.

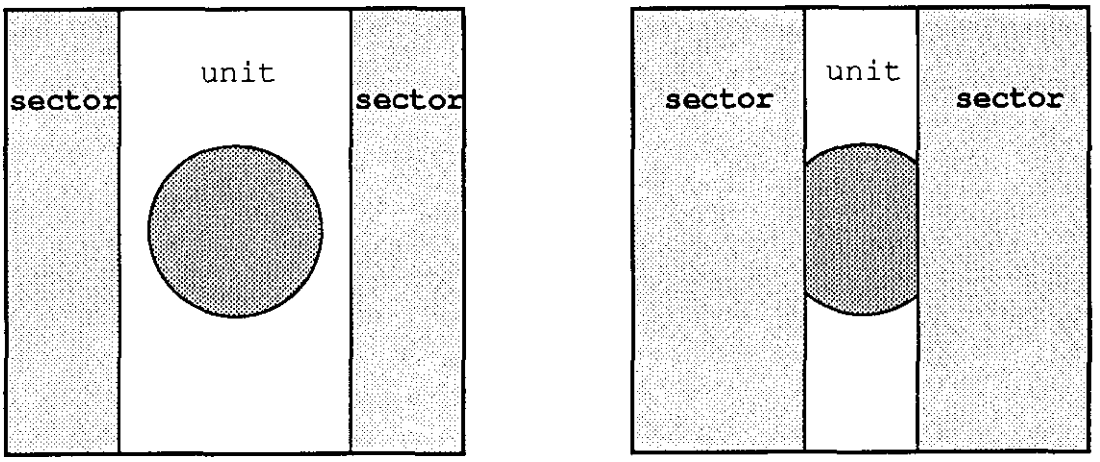
Note: When you write a unit statement , such as:

```
UNIT  5  AA5  +17  -23  +39
```

the surfaces 17, 23, and 39 determine the outer boundaries and, consequently, the size, shape, and location of the unit. When unit 5 is defined, the surfaces used in its definition define a set of sectors of a given size and shape and location. As a particle moves in, say level-0, and encounters a unit, the position of that particle is transferred to the unit definition at level-1. That position may be in

the middle of the defined unit or it may be outside the surfaces used and in a fill sector.

The two sketches below illustrate the same defined unit used in two cases where the unit statements leave different size holes for the unit. Both cases will run. You must insure that the resulting physical model is what you wanted.



Translation and Rotation

Now, if we are going to write several unit statements each using the same UNIT definition, we will run into a problem. The surfaces used within the UNIT definition have a common coordinate system. A sphere located at (3,–7,2) is at a fixed place in space. The unit statement that calls a definition including this sphere will have to be in a physical location that includes the volume around (3,–7,2). If another unit statement employs this same definition, it also should include the volume around (3,–7,2). The two volumes can not be at the same place, however. It is necessary then to be able to do a translate–rotate operation similar to that done when defining surfaces.

Assume that the surfaces written to be used within a UNIT definition exist in an (x",y",z") coordinate system (these surfaces may already have individual TR commands included in their definitions in the SURFACES input block). When writing the unit statement, we will include a TR definition to relate this coordinate

system to that used at the level in which the unit statement is written. That statement is now of the form:

UNIT *number name* $\pm s_1$ $\pm s_2$ ... (TR x_0 y_0 z_0)
(x_1 y_1 z_1) (x_2 y_2 z_2)

where (x_0 y_0 z_0) is the center of the (x",y",z") coordinate system expressed in terms of the coordinate system used at the level from which the UNIT statement is written. (x_1 y_1 z_1) is a point on the positive x" axis, but not the origin, and (x_2 y_2 z_2) is a point on the positive y" axis.

Now a single UNIT definition may be moved from one place to another and used in many different geometric positions. The basic unit may also be used with a different orientation in each of the positions in which it is employed. Note that the TR command does not change the surfaces used in the unit statement—i.e., the UNIT *number name* $\pm s_1$ $\pm s_2$ ....

The rotation of a unit allows a user to set up some problems in a simple fashion and then move them into a position that would have been difficult to describe. Take, for an example, a series of geometric surfaces related to a single axis. These could be boxes, cylinders, or other surfaces all parallel or perpendicular to this axis. If the axis is along one of the coordinate axes, there is little trouble in defining the needed surfaces. However, if the axis is not in one of the three simple directions, it could take considerable effort to specify the needed surfaces. The unit definition allows you to define this part of the geometry using simple directions. You then specify the unit with the appropriate TR statement, used only once, to force the axis into the correct direction.

Like the similar TR statement used with surfaces, the absence of any TR input implies no translation or rotation; the existence of just the first point implies only translation with the new axes parallel to the original; and the existence of the first two points implies that the direction of the y"- and z"-axes are unimportant to the definition of surfaces used within the unit. When using the OR option within a unit statement, the appropriate TR statements must be given with each OR input group.

Defining the Geometry

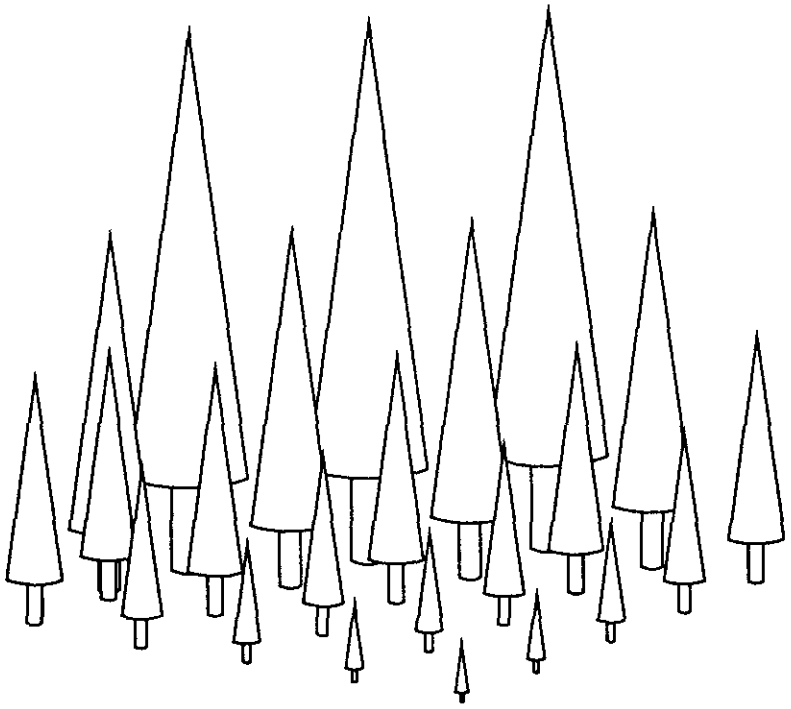
Scale Factors

In a few very rare cases, it is desirable to use the same UNIT definition in a number of places and to vary the size of the defined unit in each place. The full UNIT definition then becomes:

```
UNIT  number name ±s1  ±s2  ...  (TR x0 y0 z0)
      (x1 y1 z1)  (x2 y2 z2)  (SF=scale-factor)
```

where the *scale-factor* is any positive number greater than zero. All surfaces used within the problem will be altered by this scale factor. A point that is at (*x, y, z*) in the defined geometry will now appear to be at (*sf*x, sf*y, sf*z*). This causes not only the relative size of the defined figures to change in size, it also moves the surfaces. The only stable point is the coordinate center. It may be desirable to make this the center of the unit being described.

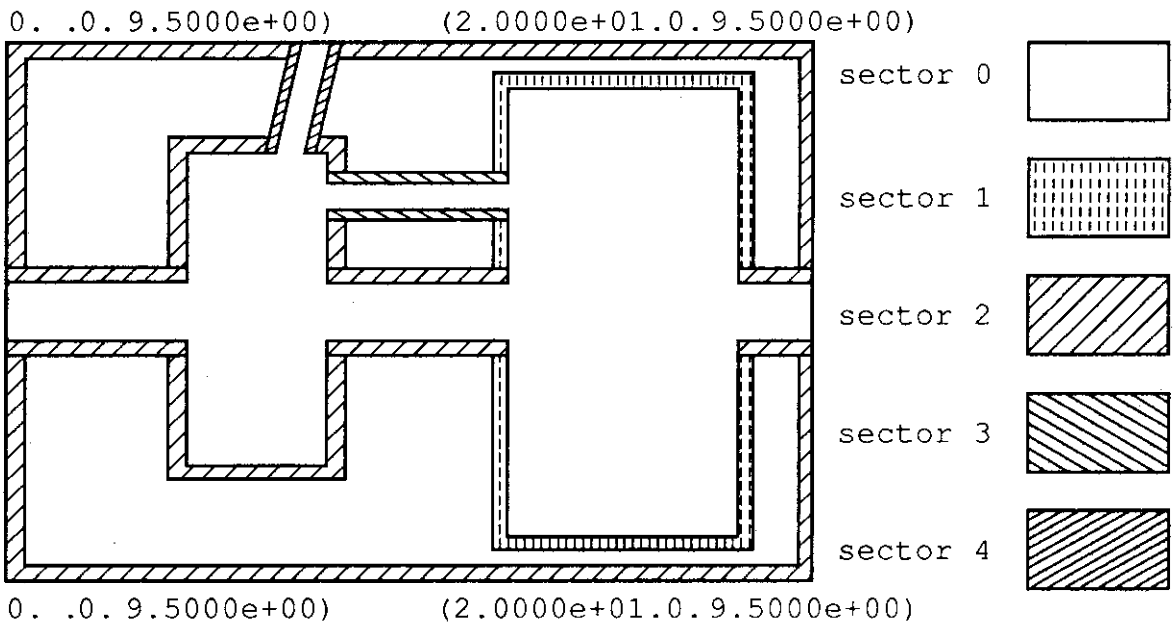
The illustration below shows a geometry where there is only one unit definition—a very simple pine tree—used many times with varying scale factors.



Checking Your Input

Cross-Section Pictures

When initially describing the geometry of a problem, it is usually desirable to look at pictures that present the COG understanding of your inputs. Many errors can thus be detected and corrected before excessive amounts of computer time are consumed. The quickest form of picture is the cross-section cut. For a specific part of a plane within the geometry, which the user provides, the code will draw a picture that will represent the boundaries between materials, regions, or sectors. The area associated with each is identified by a distinctive pattern. The illustration below is from a COG problem.



Defining the Geometry

To obtain a cross-sectional picture, input the following card after all sectors, boundaries, units, ... have been specified in the GEOMETRY block of data:

```
PICTURE  [          ]  [ SECTOR  or  SEC  or  S ]   xt1  yt1  zt1
         [ CS       ]  [ MEDIA   or  MED  or  M ]
         [ SECTIONAL]  [ REGION  or  REG  or  R ]

   xb1  yb1  zb1  xbr  ybr  zbr  {TITLE = "...."}
```

where:

the SECTOR, MEDIA, or REGION (or the shorter substitutes) defines what is to be shown in the pictures. Point *t1* will be at the top left corner of the picture, point *b1* at the bottom left corner, and point *br* at the bottom right corner. The corner described by these three points must form a right angle. The picture scaling will be the same in both directions and will be picked from these input points. The title input is optional and will be added, if provided, to the title of the problem to assist you in identifying this particular picture.

When drawing pictures of this type, the code goes out of its way to look for input errors. These are tagged and a full error print is output to assist you in determining the error. In the section **Error Messages** below (page 96), this output is illustrated. At this point, the work on this particular picture is terminated but subsequent cross-sectional pictures will be attempted. No other type of picture will be drawn, and volume calculations will not be undertaken. The code will also read the remaining input data but will not enter the random-walk phase of calculations.

Each desired picture requires its own input starting with the word PICTURE.

All cross-section pictures based on sector data will be drawn before those based on media or region data. The cross-section pattern for a given number sector will be the same in the entire set of such pictures and will be illustrated on a printed key.

Perspective Black and White Pictures

A perspective picture is the kind you would get if you stood back and took a photograph of your geometry. The black and white version illustrates with lines the edges and boundaries between sectors (or materials or regions). A given input list specifies which sectors, etc., are visible. The rest are invisible. Thus, a picture can illustrate just part of the geometry, and one can look *inside* to see what is there.

Perspective pictures are requested by:

```
PICTURE [ P           ]  [ SECTOR  or  SEC  or  S ]  xc  yc  zc
        [ PERSPECTIVE ]  [ MEDIA   or  MED  or  M ]
                         [ REGION  or  REG  or  R ]

    r  d  θ  φ  s1  s2  s3  ....  {TITLE = "...."}
```

where:

x_c, y_c, z_c is the location in the defined geometry of that point that will occur in the center of the picture.

r defines the radius of a sphere centered on this point. Everything within the sphere will be included in the picture.

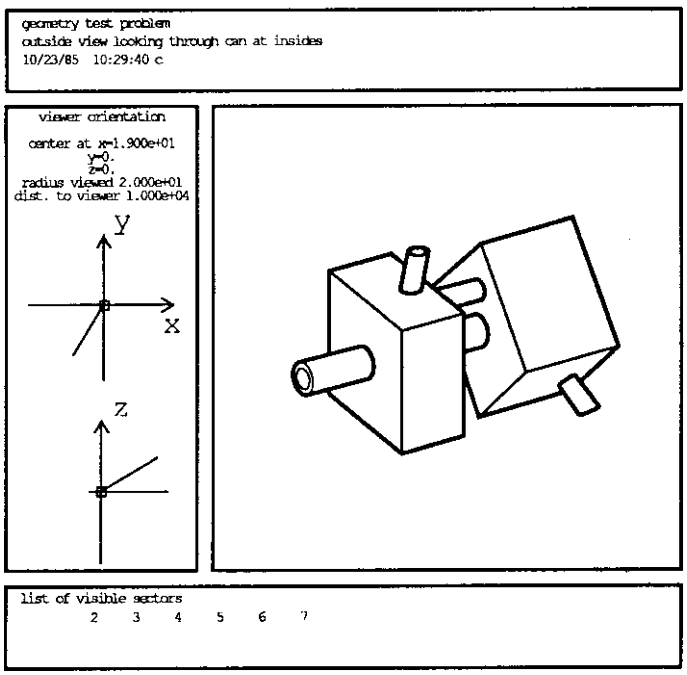
d, θ, ϕ are spherical coordinates identifying where the viewer will stand to look at the geometry. d is the distance of the viewer from the center of the picture, θ is an azimuthal angle measured in a plane parallel to the x,y plane and passing through the picture center. θ=0 is a line in this plane parallel to the +x-axis and passing through the picture's center. θ=90 is a line in this plane parallel and going in the direction of the +y-axis. ϕ is measured upward from this plane toward the +z-axis. Angles are expressed in degrees.

s_1, s_2, ... is a list of sectors (or materials or regions) that the viewer can see. Any sector that is not in this list is transparent.

Again, each picture requires one PICTURE input.

Defining the Geometry

The following figure illustrates such a picture drawn by the code.



The routines that draw these pictures attempt to do as little work as possible. The picture is broken up into a uniform mesh and the *visible* sector and its associated surface at each mesh point is determined. If the four corners of a cell of this mesh all *see* the same sector, it is decided that no lines need be drawn within this cell. Sometimes this does not work and there results a blank square on the picture. This can be solved if you:

- Use your pencil to fill in the missing lines.
- Shift the picture just a little by moving the center or one of the viewing angles and then letting the code try again.
- Look at this specific point in much closer detail by picking a whole new set of picture locations and redrawing the new picture.
- Replace `P` or `PERSPECTIVE` with `P*` on the picture statement and then redrawing the picture. This forces the picture mesh to have twice the number of points.

Like cross-sectional pictures, if an error in the geometry is determined, the picture drawing will stop at that point and an error printout will be given. Other cross-sectional pictures will be attempted, but no color pictures or volume calculations will be performed. Only the input phases of COG will be completed before code termination.

Perspective Color Pictures

Perspective color pictures† are not debugging aids but were included so that COG users could produce slides or color prints for use in meetings or reports. They should be produced only after the geometry has been debugged and after similar black and white pictures drawn.

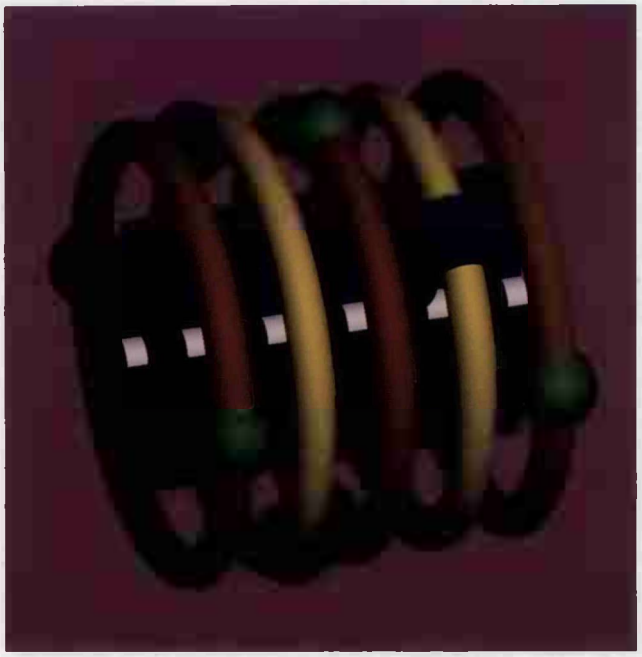
The code inputs are identical to the black and white perspective pictures except that the P or PERSPECTIVE or P* is replace by COLOR or C. The output for these pictures will be three files written by the code and left in memory at the conclusion of a problem run. These must be given to the Dicomed machine in the appropriate order and format. There are post processors to do this and you can get these programs from the COG programming group.

There is a standard set of colors built into the code that offers a good variation between the different sectors. The default standard does not include the casting of shadows. The COLORSET block of data that alters the defaults and allows control over many features of the final picture are described in **Special and Rarely Used Items** (see page 198). Before altering the standards, it is advisable to talk to others who have used these options and also to have some idea of the desired colors and effects.

† We are indebted to Alan Christiansen, a summer employee in 1983, who did the initial work on this option.

Defining the Geometry

A color picture obtained from the code is reproduced below.



Volume Calculations

Another option available for debugging the geometry description of a problem is the volume calculation. In this option, you specify a box-shaped volume within your geometry and the code performs a Monte Carlo calculation of the volume and mass of each material, region, or sector within that box. This can be an expensive calculation, but it is also the best method for finding errors associated with overlapping sectors and, when you compare the results to the original system properties, to the volume or mass of individual components.

The method is employed when the following input is inserted into the GEOMETRY block of data:

```
                 [ SECTOR  or  SEC  or  S ]
VOLUME           [ MEDIA   or  MED  or  M ]   x0 y0 z0   x1 y1 z1   x2 y2 z2
                 [ REGION  or  REG  or  R ]

length-x'  length-y'  length-z'  {TITLE = "...."}
```

where:

A volume request starts with the word VOLUME and follows the following description:

- x_0, y_0, z_0 is a reference corner of the box expressed in the level-0 coordinates.
- x_1, y_1, z_1 is any point along one edge, or its extension, of the box. This, along with the reference corner, defines the +x'-axis.
- x_2, y_2, z_2 is any point along another edge, or its extension, of the box. This, along with the reference corner, defines the +y'-axis. The z'-axis is defined using a right handed system.
- *length-x'*, *length-y'*, and *length-z'* are the length of the defined box along each of the three axes.

Each separate volume request starts with the word VOLUME and follows the description given above.

Defining the Geometry

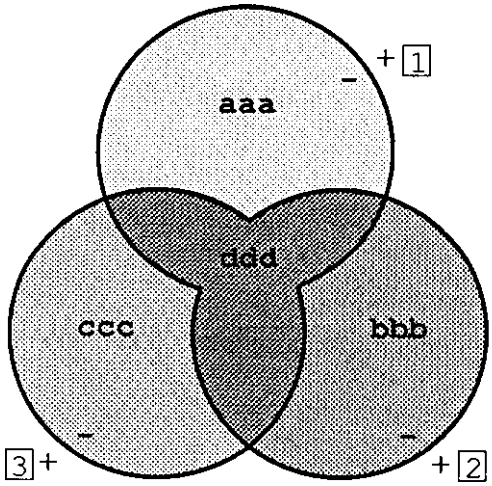
The following is an example of the output from a volume calculation.

```
VOLUME CALCULATION

        point on corner (orgin)    ( 6.00000e+00, -6.00000e+00, -6.00000e+00)
        point on x'-axis           ( 7.00000e+00, -6.00000e+00, -6.00000e+00)
        point on y'-axis           ( 6.00000e+00, -5.00000e+00, -6.00000e+00)
        z'-axis forms a right handed system with above
        length of x' side =   6.00000e+00
        length of y' side =   1.20000e+01
        length of z' side =   1.20000e+01

        media number          volume      fsd          mass (kg)     fsd
              1             2.543e+02   0.007          2.543e+02   0.007
```

Error Messages

COG is able to detect many different kinds of input errors while reading the geometry. These are printed out in a form that should be self-explanatory. While drawing pictures or calculating volumes, the code is sensitive to an overlap of sectors. If this occurs, a special message is printed that gives all the information that the code knows including the boundary just crossed (if any) and the relationships to all surfaces in current use. With this information, and a lot of time and thinking, the user should be able to identify the problem with its defined geometry.

To illustrate the error procedure, the following case was run. Note that the definition of sector 3 missed a reference to surface number 2.



BASIC GEOMETRICAL DESCRIPTION

	no.	name	med	reg	df	--surface relationships--		
SECTOR	1	aaa	1	1	1.000e+00	-1	+2	+3
SECTOR	2	bbb	1	2	1.000e+00	+1	-2	+3
SECTOR	3	ccc	1	3	1.000e+00	+1	-3	
SECTOR	4	ddd	1	4	1.000e+00	-1	-2	
					or	-2	-3	+1
					or	-1	-3	+2

The following error was obtained.

```
ERROR  --  point found which is in al least two sectors
           given point in level 0 coordinates
               x =  1.49175349e+00
               y = -6.17806995e-01
               z =  1.00000000e-03
           direction cosines
               u =  1.00000000e+00
               v =  0.
               w =  0.
           relation to each surface at this point
                 equation    evaluation
                     1            +
                     2            -  (just crossed this surface)
                     3            -
           numbers and names of sectors which contain this point
                        3       ccc
                        4       ddd
```

Tie the Geometry Input to the Rest of the Problem

Relating Material Media Numbers, Statistical Region Numbers, and Density Factors to Sector Numbers

The previous sections defined specific surfaces by giving them a definition along with a defining number. Specific pieces of volume, referred to as sectors, were then defined in terms of their relationship to the surfaces. Both surfaces and sectors exist within the geometric description of a problem. It is now necessary to connect the sectors to the rest of the problem. In doing so, we make use of three additional quantities—material media, density factor, and statistical region.

Material media. Material media are physical materials (such as water or steel) that exist in one, or more, of the sectors. Each is identified with a specified positive integer assigned by the user. The MIX block of input, defined in detail in **Defining Material Media**, below, specifies the composition of each material in terms of the physical density of each element (or isotope) that forms the material.

Reference to a medium with a number equal to zero implies a void, and reference to one with a number equal to minus one implies a fictitious material with an infinite absorbing cross section. These two special cases do not have to be further defined by the user. Any set of material media numbers may be employed in a problem, but each used medium must be specified. In the inputs below, the letter *m* is used to signify a given media number.

Density Factor. Some problems exist that have the same material in a number of different sectors, but the density of the material is different in each. An example might be a reactor in which water is boiled and converted to steam. The density of water then could be different in various parts of the reactor according to the amount of steam present.

It is convenient to talk about the density of the material with relation to the density of the same material at some given point. This is what is defined as the density factor. The defined material in the MIX block inputs has a relative density of one. In any sector

where this same material is used, the density factor is then the density of the material in the sector divided by the density that was used in the MIX block. The inputs below indicate this quantity by *df*.

Statistical Region. A statistical region identifies a certain geometric volume of the problem in which the user wishes to use the same random-walk modification parameters or to tabulate the events that occur.The statistical region can also be used to define detectors that provide the required output results.

Each region is, again, identified with a positive integer provided by the user. The WALK set of input blocks, the ANALYSIS block, and the DETECTOR block make extensive used of region numbers. The examples below refer to the statistical region by the letter *r*.

If no other specification is made in the input, the media and statistical region numbers are set to equal the sector number and the density factor is set to one. The exception to this is a medium of –1 or 0, both that will result in a region number of zero.

When the user wishes to be more specific in relating sectors, media, regions, and density factors, there are seven different blocks of input data that allow the default arrangement to be changed. Any, all, or none of these blocks may be employed as the user sees fit. However, only one block of a given type may be used at a time. If the relationship specified by these inputs is changed more than once, the physically last block in the input definitions will be used.

Defining the Geometry

The ASSIGN Blocks

ASSIGN. The ASSIGN block specifies the full relationship of media, region, and density for each sector.

```
ASSIGN  s1 m1 r1 df1     (s2 m2 r2 df2)
        (s3 m3 r3 df3)   ..................
```

ASSIGN-M. The ASSIGN-M block simply reassigns the media number of the specified sectors.

```
ASSIGN-M  s1 m1   (s2 m2)   (s3 m3)   .......
```

ASSIGN-R and ASSIGN-D. Likewise, the ASSIGN-R and ASSIGN-D blocks give the region and density factors for specified sectors.

```
ASSIGN-R  s1 r1   (s2 r2)   (s3 r3)   .......

ASSIGN-D  s1 df1  (s2 df2)  (s3 df3)  .......
```

ASSIGN-MD. The ASSIGN-MD block gives both media and density values for the selected sectors.

```
ASSIGN-MD  s1 m1 df1  (s2 m2 df2)  ........
```

ASSIGN-ML and ASSIGN-RL. The two methods, ASSIGN-ML and ASSIGN-RL provide a list of sectors to go with a specified media or region.

```
ASSIGN-ML  ma  s1a  s2a  s3a     ......
           ( / mb  s1b  s2b  s3b  ...)     ......

ASSIGN-RL  ra  s1a  s2a  s3a     ......
           ( / rb  s1b  s2b  s3b  ....)    .......
```

Note: The slash (/) is an input and *must* have blanks on both sides of it.

Data Input Example Using ASSIGN Blocks

As an example, let us look at a problem with four sectors. We will change these around in a very unrealistic way just to illustrate the use of ASSIGN statements. Starting with the original sector definitions we would have:

	Sector 1	Sector 2	Sector 3	Sector 4
Media	1	2	3	4
Region	1	2	3	4
Density factor	1.0	1.0	1.0	1.0

With the following ASSIGN input,

```
ASSIGN  1  2  1  1.5    3  2  1  0.75
```

these would change to:

	Sector 1	Sector 2	Sector 3	Sector 4
Media	2	2	2	2
Region	1	2	1	4
Density factor	1.5	1.0	0.75	1.0

And by adding,

```
ASSIGN-M   2  5    3  6
ASSIGN-D   4  2.7
```

we would then have:

	Sector 1	Sector 2	Sector 3	Sector 4
Media	2	5	6	4
Region	1	2	1	4
Density factor	1.5	1.0	0.75	2.7

Defining the Geometry

Then adding,

```
ASSIGN-RL  1  1  2  3  /  2  4
```

the definitions would stand as:

	Sector 1	Sector 2	Sector 3	Sector 4
Media	2	5	6	4
Region	1	1	1	2
Density factor	1.5	1.0	0.75	2.7

and the addition of:

```
ASSIGN  2  2  2  1.0
```

would produce a fatal input error since one ASSIGN block of input has already been specified in the input.

Defining Material Media

The MIX block of input specifies a physical material to go with each material number the user enters. The physical material is defined by one or more component elements, isotopes, or mixtures with their associated specific gravities. Within COG, this information allows the required cross-section data to be extracted from one of the disk-file libraries and then used in determining the random walk of particles.

A number of disk files are used with cross-section inputs—for example, one contains a dictionary of allowable input names and mixture definitions, and others contain physical cross-section data. A default value for each allowable input name is built into COG, and default files to go with each default value exist in the public files on the LLNL Cray machines.

Neutron data in the default files represent materials at room temperatures. It is possible to generate other files that contain different evaluations of cross sections or data at other than room temperatures. If this is needed, contact the COG programming group for further explanations on how to obtain the required data and libraries.

Changes from the default values are made by providing, at the beginning of the MIX block, one or more statements of the following type:

```
SYMBOL = filename
```

where SYMBOL is a name from the list below and *filename* is the new file name to be used for this information. Needless to say, a file by that name must exist at the time the problem is run.

Data type	SYMBOL	Default name
dictionary	LEX	COGLEX
neutron	NLIB	COGNXS
thermal neutron	SABLIB	COGSAB
proton	PRLIB	COGPXS
deuteron	DLIB	COGDXS
tritium	TLIB	COGTXS
He-3	HLIB	COGHXS
alpha	ALIB	COGAXS
photon	PLIB	COGGXS
electron	ELIB	COGEXS
positron	POLIB	COGFXS

The user should understand that the information provided in the default-value files is an input into COG even if the input is done automatically. In general, the cross-section data are taken from the ENDL compilation as documented by the UCRL-50400 series of publications or from the ENDF/B evaluations.

Results obtained from calculations using these data are only as good as the information itself. The knowledgeable user will have acquired a *feeling* for the cross-section information and its accuracy and the consequent accuracy of his calculated results. To assist the user in this process, COG will plot the total cross section of each defined material. Other routines exist external to COG that will list all or part of any cross-section library.

Materials are defined as follows:

```
MAT=material-number  (TEMP=physical-temperature)
component-1 specific-gravity-1  (component-2
specific-gravity-2) (component-3  specific-
gravity-3)  ..........
```

Defining the Geometry

The temperature is the physical temperature of the material in degrees Celsius. If omitted, this is set automatically to 20°C. For cases where one or more materials have temperatures not equal to 20°C, the neutron cross-section libraries must include data for doppler-broadened resonances and thermal scattering at the requested values. This implies the use of nonstandard neutron libraries.

Each component is the name of an element, isotope, or mixture defined in the cross-section library. In Appendix A (page 253) are lists of the contents of the dictionary and the availability of data for both neutrons and photons.

It should be noted that each element, isotope, or mixture has associated with it a density specified by its specific gravity. For all practical purposes, this is the density in grams/cc. Because the input units to the code can be altered by specifications made in the BASIC block and because it was stated that all such specifications would alter the units used throughout all the problem inputs, the required density could end up as units such as grams per cubic mile. Specific gravity does not change with the units, thus its use.

As an example, assume that material media number 7 is silicon dioxide with a density of 2.64 gm/cc. The silicon density, or specific gravity, would be:

$$(2.64)(28.09) / (28.09 + 32.00) = 1.23$$

and that of oxygen would be:

$$(2.64)(32.00) / (28.09 + 32.00) = 1.41$$

Assuming that the material is at room temperature, the input to COG would look like:

```
MAT=7  SI  1.23    O  1.41
```

Because users have different ideas about what symbols should be used to represent elements, a number of different choices are available. If you consult Appendix A, you see that you could use any of the following for silicon:

```
SI
14
14000
SILICON
```

And, for oxygen:

```
O
O16
8
8000
8016
OXYGEN
```

The 8016 or O16 input signifies that natural oxygen is entirely composed of the isotope with a weight of sixteen. In Appendix A you can also see that both neutron and photon data are available for each of these two elements.

Now, if you look further in Appendix A, you find that some entries represent mixtures. On page 276 (page 24 of the listing in this appendix), silicon dioxide is given. Rather than calculate each component density, we could specify material 7 as:

```
MAT=7    SIO2   2.64
```

and COG would realize that there were two elements in this material and calculate the correct density for each of them.

On closer examination, we will realize that some neutron data for elements are made up of component isotope data. Thus, lithium is a composite of Li-6 and Li-7. Other mixtures involve a fairly large number of elements. The general purpose of having such mixtures in the library is to save the user the time required for hand calculations. However, the compositions given in the dictionary may not always be correct for all problems. As an example, concrete is different in different regions, because the composition of the sand and gravel used to make it varies from region to region. It is even different in one area, because slight variations occur in the relative amounts of sand, gravel, lime, and water. It is even different within the same batch, since the local temperature during setup varies due to heat transfer; this

variation causes a variation in the amount of retained water. Thus, no one *concrete* correctly represents the composition of all concretes. Those compositions given in the library are representative concretes and should be used with care. The same statement is true for other compositions built into the dictionary.

In the same way, trace amounts of elements are not included in the standard definitions. For most cases, this does not affect the calculation results. In other cases, the result is attributed only to reactions occurring in the trace elements, and large errors could result by using the *standard* definition in the library. The classic example of this is the thermal neutron activation of structural aluminum. The aluminum itself does not activate, so all the resultant activation comes from the less than 1% impurities. Again, the user must exercise a certain amount of expertise and not just blindly use the built in mixture definitions.

It should be noted that the densities given in Appendix A are not used by COG and may not be correct in all cases. They exist in the COG dictionary, because this same dictionary is used with other codes that do require these data.

Neutrons that reach thermal energies undergo elastic scattering represented by calculations using the heavy-gas model. The following mixtures are exceptions to this general statement:

WATER or H2O
HVYWATER or D2O
POLYETHY or CH2
BENZENE or C6H6
BE-METAL
BEOXIDE or BEO
GRAPHITE
ZRH2
UO2 or U233O2 or U234O2 .... or U240O2

In these cases, the ENDF/B S(α,β) tables are used to calculate the appropriate exit energies and directions. Note that water defined by the user as the combination of hydrogen and oxygen will use scattering from the heavy gas model while that defined as the single mixture WATER will use the S(α,β) tables.

A typical MIX block input would look like:

```
MIX  MAT=1  WATER  1.0    MAT=2  U23802  18.5
MAT=5  N  0.00125
```

If this was at 80°C with different libraries, these inputs would be:

```
MIX  NLIB=CNXS80  SABLIB=CSAB80  MAT=1  TEMP=80
WATER  1.0

MAT=2  TEMP=80  U23502  18.7  MAT=5  TEMP=80  N
0.00125
```

Time-Dependent Geometry Specifications

It is sometimes necessary to simulate a geometry that moves with time. Within the nuclear test program, problems like these are encountered where a collimator in a line-of-sight has sufficient energy deposited in it to cause it to close. A time-dependent geometry must therefore be specified to obtain the transmission down this line-of-sight properly.

COG simulates a time-dependent problem by accepting the description of the geometry during each of a series of time *snapshots*. The geometry description remains constant during each single snapshot. When a particle reaches a point at a time when the snapshot changes, it is held stationary in position, and the new geometric description is entered. The sector location of the particle in this new geometry is determined, and the particle then continues along its original trajectory. Obviously, the more snapshots you use in a problem, the closer to the correct value the final calculated results will be. It should be stressed that COG does not calculate by itself the movement of the surfaces. This work must be performed by other methods external to COG.

To activate the time-dependent option in COG , use several GTIME blocks of input information, each followed by their associated GEOMETRY, SURFACE, and ASSIGN blocks. There is only one piece of information provided in a time block, t, the largest time for which the data following this card is to be used. A typical time-dependent

problem would then have blocks of input information that would look like:

```
BASIC
GTIME  t1
  GEOMETRY
  SURFACE
  ASSIGN
GTIME  t2
  GEOMETRY
  SURFACE
  ASSIGN
GTIME  t3
  GEOMETRY
  SURFACE
  ASSIGN
.........
MIX
SOURCE
ANALYSIS
DETECTOR
.........
END
```

The ASSIGN block represents any of the forms of assignments that can be input. The first set of descriptions, given after the GTIME t_1 and before the GTIME t_2 inputs, are used from t=-10^{99} to t_1. The second set is used from t=t_1 to t=t_2, etc. Obviously, the input times must be in increasing order. Note that only the GEOMETRY, SURFACE, and ASSIGN inputs are time dependent. The MIX block must describe all mixture compositions used in all time snapshots and the SOURCE block must describe the source over all problem times.

The user should be aware that in the summary of random-walk events it is possible for a particle, at the time the boundaries are switched between snapshots, to move into or out of a region without being properly counted. The resulting imbalance provides to the user some measure of how frequently this event occurs.

Specifying the Fixed Source

General Method of Source Specifications

A problem without a source of particles has zero fluxes throughout all parts of phase space. Therefore there must be a finite nonzero source specified in each problem. It is up to the user to give the characteristics of this physical source as accurately as possible. It should be remembered that a problem run with an incorrect source will produce an answer, in some cases a seemingly good answer, but it is the answer to the wrong problem. Please exercise care to insure that the source is physically correct.

Most source specifications are simple and can be input to COG with a few easily understood statements. There occasionally arises, however, rather complex source specifications. These take more inputs and greater amounts of the user's time. The full use of all of the descriptions given in this chapter occurs only for these few cases. The user should not be scared off by the large number of options available.

A source description is given by the position of the particle in space, the direction of initial movement, the energy (including the type of particle), and the time at which the particle starts—that is:

$$S(\vec{r}, \vec{\Omega}, E, t)$$

In most cases, the dependence on each of these variables is not a function of the other variables. Thus, we often speak of a source at a given space, started isotropically, with a specific energy, and at a fixed time This ability to separate the parameters implies that the source can be written:

$$S(\vec{r}) \cdot S(\vec{\Omega}) \cdot S(E) \cdot S(t)$$

and that each of these four items can be picked independently of the rest.

Specifying the Fixed Source

Now, assume we have a source that cannot be separated this way but can be approximated by a set of increments. The ability to separate the parameters can be assumed within each increment without too much error. The source could then be represented by:

$$S_i(\vec{r}) \cdot S_i(\vec{\Omega}) \cdot S_i(E) \cdot S_i(t)$$

This is the method that COG uses. Various inputs suitable for describing each of these four variables are detailed in the sections **Position Dependence** (page 113), **Energy Dependence** (page 123), **Time Dependence** (page 127), and **Angle Dependence** (page 129) Details for putting these together to form one or more increment descriptions are in **Increment Specifications** (page 134).

Let's drop back and give two examples of increment specifications. The first is the simple case where there would be only one increment:

```
INCREMENT number S(point) S(isotropic) S(fixed E) S(fixed time)
```

where *number* is the number of particles born in the increment, and the user would have to describe each of the four separable variables.

Second, let's look at a little more complex case. In the LLNL pulsed neutron sphere experiments, the time dependence of the source is a Gaussian distribution, the position dependence is in a small volume assumed to be a point, but the intensity of the source is a function of angular direction and may be given by:

$$0.937 + 0.063 \cos\phi$$

and the energy is also related to the direction by:

$$E = 14.120 + 0.955 \cos\phi + 0.035 \cos^2\phi \ \text{(MeV)}$$

The (energy,angle) relationships are not separable. If we break the source into a number of increments in initial direction, we can

pick a single energy for use within each. The energy will be a little wrong but not too far off as long as we use enough increments. We would then have inputs as:

```
INCREMENT .00810 S(pt.) S(  0 -10°) S(15.10 MeV) S(Gaussian)
INCREMENT .02402 S(pt.) S( 10 -20°) S(15.07 MeV) S(Gaussian)
INCREMENT .03907 S(pt.) S( 20 -30°) S(15.01 MeV) S(Gaussian)
.....................................................
INCREMENT .00709 S(pt.) S(170-180°) S(13.21 MeV) S(Gaussian)
```

Both the *number* given for each increment and the angular definition terms specify the angle dependence of the source.

We could give a third example where the increments represent a spatial variation in source intensity, but by now you should have grasped the fundamentals of the method.

If the increment method just doesn't work or is too burdensome, you may make up your own set of starting source parameters and give them directly to COG. This is discussed in more detail below.

After the block name and before the variable and increment specifications, three parameters may be specified. The first, which must always be present, defines the number of source particles to be used in this problem.

```
[ NPARTICLES ]
[ NPARTICLE  ]  =  number
[ NPART      ]
```

Except in those cases where an existing source specification is used, after COG reads the various descriptions of the source, it will randomly generate the initial parameters for the specified number of source particles and store them in one or more disk files. Since COG likes to work with groups of 256 source particles at a time, it may make a few more particles than you wanted and store them away. It will not use these extra particles. One of the numbers stored for each source particle will be the starting random-number seed used to initiate the random walk for that particle. When initially set up, the file does not contain this number. It will be added during the random-walk phase of the problem run.

Specifying the Fixed Source

If this particular COG problem is to use source parameters contained on existing source files, the input includes the statement:

```
FILE = filename
```

where *filename* is the name of the file to be used. This should be the name of the first file in a family of source files. This input option would be used if you:

1. Are willing to use the same set of starting parameters used in a previously run problem

2. Wish to run the RETRACE option and follow a few previously run source particles with a full printout of everything they did.

3. Wish to run a correlated problem.

If you are running in mode (1), new starting random-number seeds will be placed into the existing file. Modes (2) and (3) will not replace the seeds. In all cases, no further source information need be specified. Needless to say, when you use an existing file the value of NPARTICLES cannot exceed the number of particles stored in the family of files.

The third possible general input is the single word:

```
CORRELATED
```

If present, a previously used source file must be specified. This turns on the option that starts each source particle with the same initial parameters and the same starting random number seed as the previous run. If the rest of the problem is identical, identical results will be obtained. The idea, however, is that the second problem has a small change from the first. Only trajectories affected by this change will be different, so the difference in results between the two problems are more meaningful than the difference in results between two uncorrelated runs. Any of the standard textbooks[18] discuss this very powerful method in greater detail.

Following these inputs, for the case where a source file is to be generated, there will be a series of DEFINE POSITION, DEFINE ENERGY, DEFINE TIME, and DEFINE ANGLE specifications. These will be followed by a series of INCREMENT specifications. All these specifications are discussed in the sections that follow.

In a few problems, you may want to force the source particles initially to scatter in one or more geometric positions. You may even want them to continue by scattering again in another geometric position. This can be accomplished by using the WALK-SOURCE instructions described starting on page 137.

Position Dependence

The definition of a position-dependent source specification is preceded by:

DEFINE [POSITION / P] = *number*

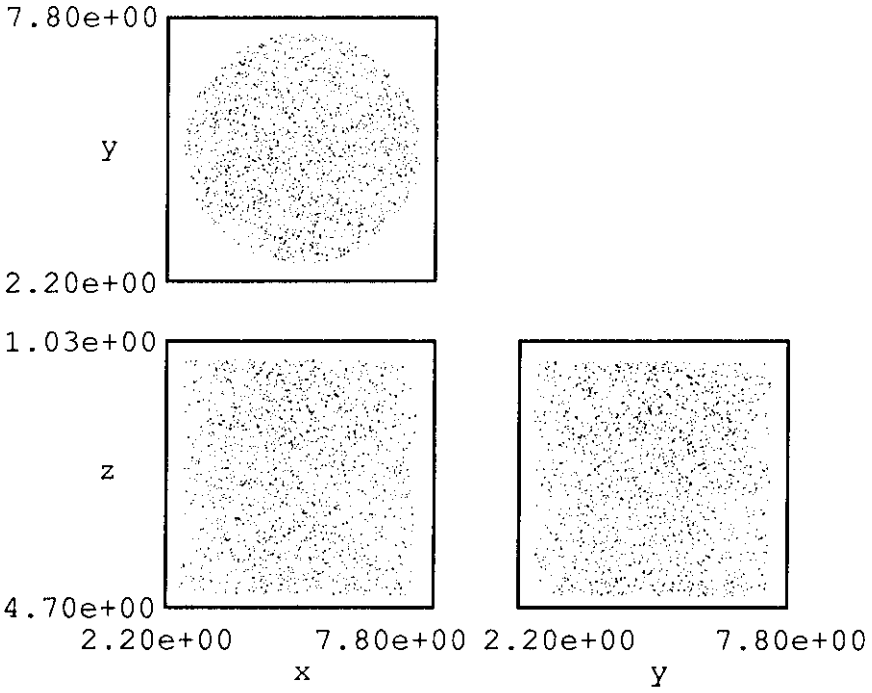
where *number* is any positive integer chosen by the user to identify this specific definition. This input is then followed by the data for one of the definitions listed below. If more than one position-dependent specification is used in a problem, the entire definition starting with DEFINE is repeated for each.

There are four classes of position-dependent sources—a point source; sources along a line, with a uniform source strength per unit length; surface sources, with uniform source strengths per unit area; and volume sources, with uniform source strengths per unit volume.

COG will read these specifications in the input phase of the code operation. COG will then present a visual representation of the location of 2000 points randomly chosen from the requested position-dependent source. The user should visually check this output to insure that the code has properly defined and positioned

Specifying the Fixed Source

the source. An example of the output for a volume cylindrical source is shown below:



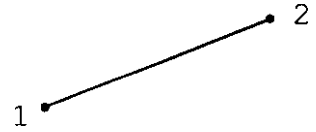
Point Source

[POINT / PT] *x y z*

- where the point is located at (x, y, z). The values of these coordinates are consistent with the level-zero geometry description and in the units of position chosen in the BASIC block; or, if not respecified, the values will be in centimeters.

Line Sources

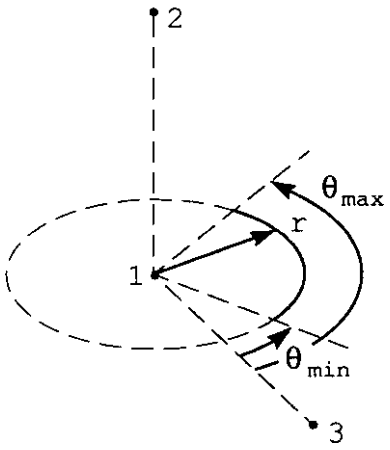
Straight-line Segment



[LINE / LIN] $x_1\ y_1\ z_1\ \ x_2\ y_2\ z_2$

where the line runs from point 1 to point 2.

A Circle or an Arc of a Circle



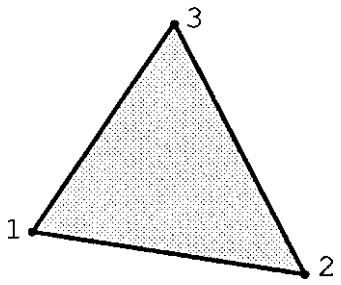
[CIRCLE / CIR] $x_1\ y_1\ z_1\ \ x_2\ y_2\ z_2\ \ r$

$(x_3\ y_3\ z_3\ \ \theta_{min}\ \ \theta_{max})$

where point 1 is at the center of the circle and point 2 is on the normal to the plane of the circle that goes through the center of the circle. The radius is given as r. If desired, an arc may be specified by giving a third point, located in the plane of the circle, to define a zero-angle direction and then providing two angles, in degrees, to specify the beginning and end of the arc. The angles are measured as shown in the figure to the left.

Surface Sources

Triangle



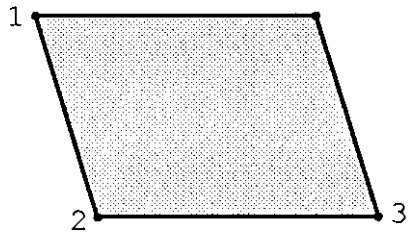
[SS-TRIANGLE / SS-TRI / SS-T] $x_1\ y_1\ z_1\ \ x_2\ y_2\ z_2$

$x_3\ y_3\ z_3$

Where the three points are at the corners.

Specifying the Fixed Source

Parallelogram



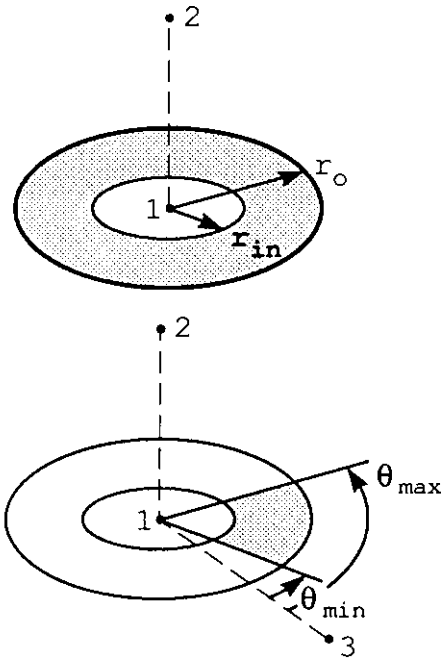
SS-PARALLELOGRAM
SS-PAR
SS-P

$x_1\ y_1\ z_1\quad x_2\ y_2\ z_2$

$x_3\ y_3\ z_3$

Where the three points are at the corners and are arranged as shown to the left.

Disk



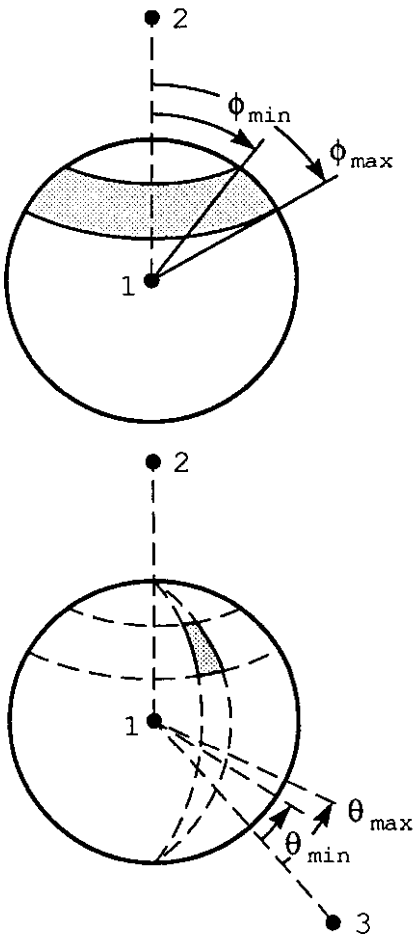
SS-DISK
SS-DIS
SS-D

$x_1\ y_1\ z_1\quad x_2\ y_2\ z_2\quad r_o$

$(r_{in})\quad (x_3\ y_3\ z_3\quad \theta_{min}\quad \theta_{max})$

Point 1 is the center of the disk and point 2 is on the normal to the disk passing through the center of the disk. r_o is the outer radius of the disk. If desired, r_{in} may be specified, thus restricting the source to an annulus. Further restriction may be obtained by giving point 3 on the plane of the disk to define $\theta = 0$ and then providing θ_{min} and θ_{max} (in degrees) between which the source will occur.

Surface of a sphere



[SS-SPHERE
SS-SPH
SS-S] x_1 y_1 z_1 r

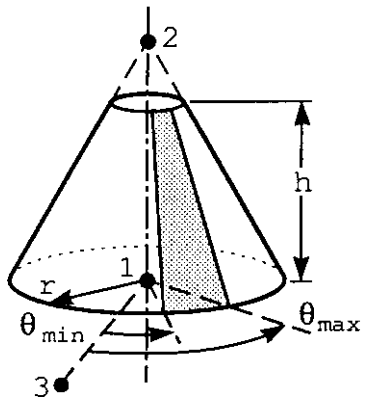
(x_2 y_2 z_2 ϕ_{min} ϕ_{max})

(x_3 y_3 z_3 θ_{min} θ_{max})

The center of the sphere is at point 1 and the radius is equal to r. The sphere can be restricted to that surface lying between two polar angles by giving point 2 so that the line from point 1 to point 2 defines the polar direction and then providing the two defining polar angles (in degrees). Further restriction is possible by giving a reference direction for an azimuthal angle—this is accomplished by specifying point 3 so that line 1–3 is perpendicular to line 1–2—and then giving the two defining azimuthal angles in degrees.

Specifying the Fixed Source

Surface of a cone

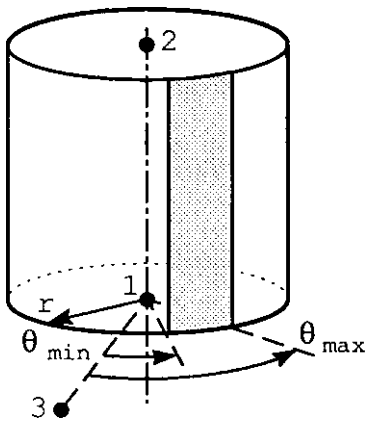


$$\begin{bmatrix}\texttt{SS-CONE}\\ \texttt{SS-CON}\\ \texttt{SS-K}\end{bmatrix} \quad x_1\ y_1\ z_1 \quad x_2\ y_2\ z_2 \quad r$$

$$(h) \quad (x_3\ y_3\ z_3 \quad \theta_{min}\ \theta_{max})$$

Point 1 is the center of the base that has a radius of r. Point 2 is the apex. If desired, h is the height from the center of the base to the center of the top of a frustum of a cone. Further restriction is possible by defining the reference direction for an azimuthal angle by specifying point 3 so that line 1–3 is perpendicular to line 1–2. The two angles (input in degrees) then restrict the surface to that part of the cone between them. Note that the surface of the cone does not include the surface of the ends.

Surface of a cylinder



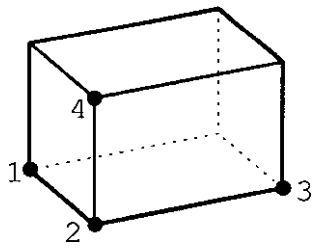
$$\begin{bmatrix}\texttt{SS-CYLINDER}\\ \texttt{SS-CYL}\\ \texttt{SS-C}\end{bmatrix} \quad x_1\ y_1\ z_1 \quad x_2\ y_2\ z_2 \quad r$$

$$(x_3\ y_3\ z_3 \quad \theta_{min} \quad \theta_{max})$$

Point 1 and point 1 are at the center of the two ends and r is the radius. If desired, the surface may be reduced by defining a reference azimuthal direction with point 3 located so that lines 1–2 and 2–3 are perpendicular and then giving the two defining angles in degrees. Note that the surface does not include the ends of the cylinder.

Volume sources

Parallelepiped



[PARALLEL | PAR | BOX] $x_1\ y_1\ z_1\ \ x_2\ y_2\ z_2$

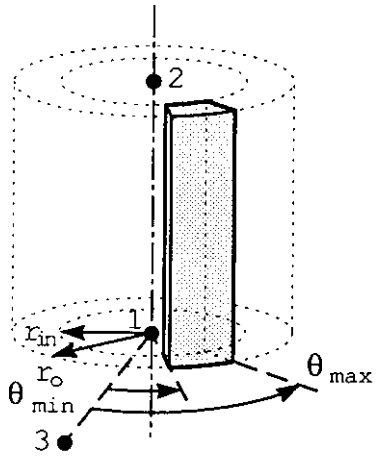
$x_3\ y_3\ z_3\ \ x_4\ y_4\ z_4$

{ [MATERIAL | MAT | REGION | REG] = *number* }

The four specified points represent four corners of the parallelepiped. Points 1, 2, and 3 are all in the same plane with point 4 above point 2 as illustrated to the left. The optional input specifies either a material or a region number. The source point that is chosen in this case must not only be within the parallelepiped but must also be at a location that contains the specified material or region.

Specifying the Fixed Source

Cylinder



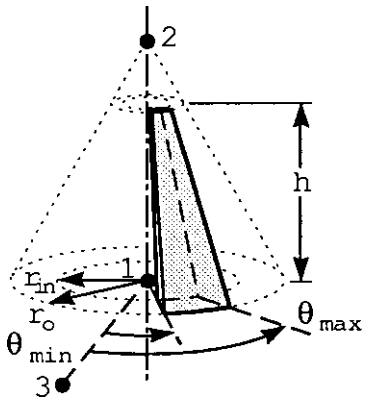
$$\begin{bmatrix} \texttt{CYLINDER} \\ \texttt{CYL} \\ \texttt{C} \end{bmatrix} \quad x_1\ y_1\ z_1 \quad x_2\ y_2\ z_2 \quad r_o$$

$$(r_{in}) \quad (x_3\ y_3\ z_3 \quad \theta_{min} \quad \theta_{max})$$

$$\{\texttt{MATERIAL} = \textit{number}\}$$

Points 1 and 2 are at the center of each end of the cylinder. r_o represents the outer radius of the cylinder while r_{in}, if input, represents the inner radius of what is then a cylindrical annulus. Further restriction of the figure can take place if point 3 and the minimum and maximum azimuthal angles (in degrees) are specified. Limits to points falling in a given material (or region) can be specified as was the case with the parallelepiped.

Cone



$$\begin{bmatrix}\text{CONE}\\ \text{CON}\\ \text{K}\end{bmatrix} \quad x_1\ y_1\ z_1 \quad x_2\ y_2\ z_2 \quad r_o$$

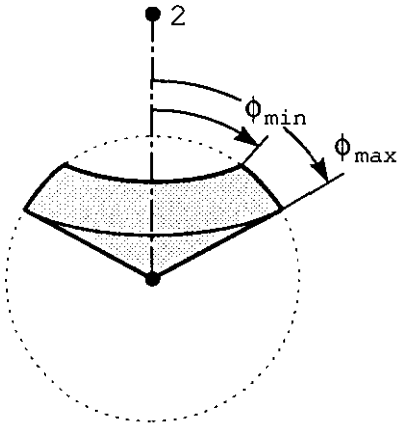
$$(r_{in}) \quad (h) \quad (x_3\ y_3\ z_3 \quad \theta_{min} \quad \theta_{max})$$

{MATERIAL = *number*}

Point 1 is at the center of the base while point 2 is the apex. r_o is the outer radius at the base. If desired, the volume may be limited to that between two cones with the same apex and center line by adding r_{in}, the radius of the inner cone on the base. Addition of h truncates the cone so that it has a height of h above the base. Further addition of a reference point in the plane of the base defines the zero azimuthal angle, and the minimum and maximum azimuthal angle (in degrees) limits the source to a segment of the cone. In the same manner as with the parallelepiped, selected points may be restricted to those falling within the defined cone and also in a specified region or material.

Specifying the Fixed Source

Sphere

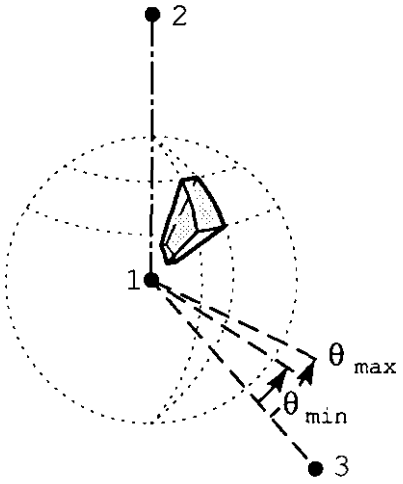


$$\begin{bmatrix}\texttt{SPHERE}\\ \texttt{SPH}\\ \texttt{S}\end{bmatrix} \quad x_1\ y_1\ z_1\ \ r_o\ \ (r_{in})$$

$$(x_2\ y_2\ z_2\ \ \phi_{min}\ \phi_{max})$$

$$(x_3\ y_3\ z_3\ \ \theta_{min}\ \ \theta_{max})$$

$$\{\texttt{MATERIAL} = number\ \ldots.\ \}$$



Point 1 defines the center while r_o gives the outer radius. The sphere may be restricted to a spherical shell by including the inner radius, r_{in}. It may be further restricted between two polar angles by including point 2 that defines the zero polar direction and then the angles. Addition of a third reference direction and two azimuthal angles cuts this last volume even further. All angles are input in degrees. Specification of a material or region number limits source points to those within the defined volume that fall in the given material or region only.

Energy Dependence

The energy dependent source specification is preceded by:

DEFINE [ENERGY / E] = *number* [NEUTRON / PHOTON / .....]

where *number* is any positive integer picked to identify this specific definition. The specification of the particle type must be present; thus, a specific definition is automatically limited to only one type of particle. This input is then followed by the data for one of the four types of energy specifications given below. If there is more than one energy-dependent specification used in a problem, the entire definition starting with DEFINE must be repeated for each.

It is possible to add to these inputs the relative importance as a function of energy. When choosing from the specified distribution, more points than normal will be chosen where the importance is high and less chosen where it is low. The statistical weight associated with each source particle will also be automatically altered, so the calculated results will be correct. If enough particles are run, the specification of importance will not alter the calculated results. If properly specified, the importance values will result in smaller statistical errors for equal calculational times. The importance may be given either as point values or as bin values as described in the section **Terminology Used to Specify Inputs** in the chapter **Getting Started** (see page 19). This implies that the specification will take the form of either:

[IMPORTANCE / IMP] $e_1\ i_1\ \ e_2\ i_2\ \ e_3\ i_3\ \ldots\ i_n$

or as:

[IMPORTANCE / IMP] $e_1\ \bar{i}_1\ \ e_2\ \bar{i}_2\ \ e_3\ \bar{i}_3\ \ldots\ \bar{i}_n\ \ e_{n+1}$

Specifying the Fixed Source

As in the position-dependence specification, a graphic representation of the energy dependence is output so the user may verify the correctness of the input. On the top portion of this presentation there are given a set of randomly chosen points picked from the distribution. It should be noted that the points so chosen include the importance specifications.

Line Specifications

Many sources that exist in nature consist of a series of discrete energy lines. Each line has associated with it a relative intensity. These many be specified by:

$$\begin{bmatrix}\text{LINE}\\ \text{LINES}\end{bmatrix}\ e_1\ intensity_1\ \{e_2\ intensity_2\}$$

$$\{e_3\ intensity_3\}\ \ldots.\ \{\text{IMPORTANCE}\}$$

Gaussian Distribution

If a source has a Gaussian distribution about a fixed energy, the code will simulate it by accepting the following data:

GAUSSIAN e_0 δe {IMPORTANCE ...}

where e_0 is the center energy and δe is the full width at half the maximum height. Internal to the code, this is simulated by a 201 point distribution evaluated from a lower energy of e_0 - 2 δe or zero, whichever is greater, to e_0 + 2 δe. The relative source per unit energy at each energy point, e, in this distribution is given by:

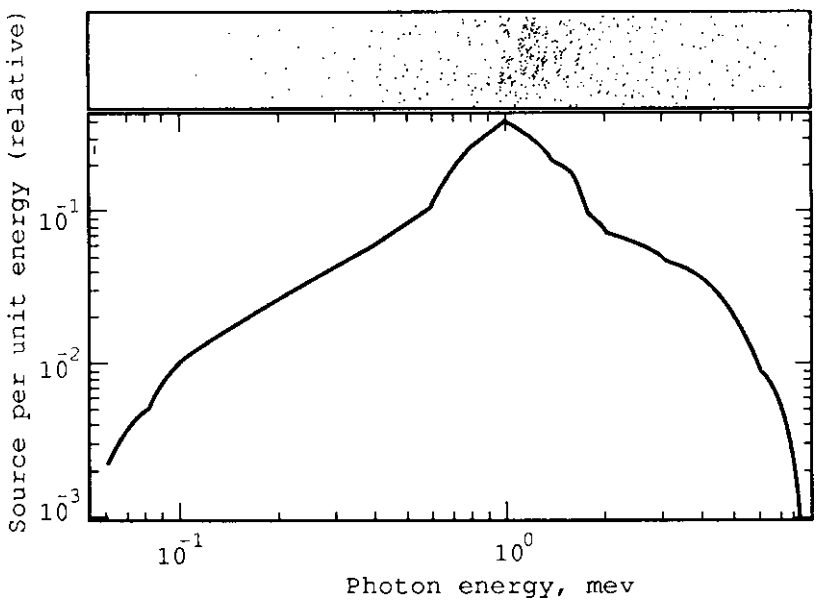
$$e^{(-.5*\ (2.35482\ (e-e_0)\ /\ \delta e)^2)}$$

Point-wise Distribution

A full distribution may be given by specifying:

DISTRIBUTION e_1 s_1 e_2 s_2 e_3 s_3 ... {IMPORTANCE ...}

where the s values represent the relative source per unit energy at the preceding energy point. It is assumed that a linear fit between adjacent energy points accurately describes the distribution. As is usual in such specifications of this type, the energies may be either ascending or descending, and the source is zero below the lowest energy and above the highest energy. An example of the printout of such an input is shown below.



Specifying the Fixed Source

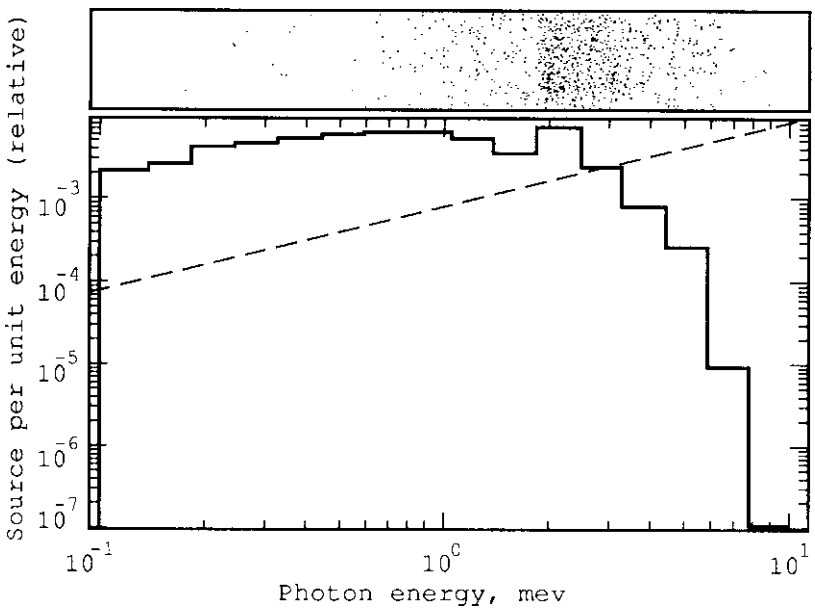
Bin Specifications

A distribution known only by averages within a given energy bin may be specified by:

BIN e_1 $\bar{s}_2$ e_2 $\bar{s}_2$ e_3 $\bar{s}_3$ ... s_n e_{n+1} {IMPORTANCE ...}

The energies may be either ascending or descending. Note the presence of the extra energy. The indicated $\bar{s}$ values are the relative average values of the source per unit energy within the appropriate bin. When choosing an energy from this type of specification, the appropriate energy group is first picked and then a specific energy is chosen uniformly between the bin boundaries. The source is zero below the lowest energy and zero above the highest energy.

The output shown below indicates the code presentation of bin data. Note that the input importance is plotted with a dotted line and that the chosen energies are those picked from the importance-weighted energy spectrum.



Time Dependence

The initial specification for a time-dependent source specification is:

DEFINE [TIME / T] = *number*

where *number* is any positive integer picked to identify this particular definition. This input is then followed by the data for one of the five types of time specifications given below. Each additional specification will also start with the word DEFINE.

Note that negative times are perfectly legal, since the zero point for time is entirely arbitrary.

As in the energy source specifications described above, it is possible to add to these inputs the relative importance as a function of time. Plots presenting the input specifications are made and are also similar to those illustrated with the energy dependence.

Delta Time Function

To indicate that all particles are born at a given time, the input specification would be:

DELTA t_0

where t_0 is the time in the units being used within the problem specifications.

If a problem increment does not specify any time-dependent specification, the default option will be used. The default is this delta function with t_0 equal to zero.

Gaussian Time Distribution

If a source has a Gaussian time distribution, it can be simulated by specifying:

GAUSSIAN t_0 δt {IMPORTANCE...}

where t_0 is the time of maximum source strength and δt is the full width at half the maximum amplitude. This is simulated by

Specifying the Fixed Source

the code by internally generating a 201 point distribution extending, in time, from $t_0 - 2\,\delta t$ to $t_0 + 2\,\delta t$. The relative amplitude per unit time is given by:

$$e^{(-.5*\ (2.35482\ (t-t_0)\ /\ \delta t)^2)}$$

Time-Dependent Point-Wise Distribution

The general distribution may be specified point-wise by giving:

DISTRIBUTION $t_1\ s_1\ \ t_2\ s_2\ \ t_3\ s_3\ \ldots$ {IMPORTANCE ...}

where the s values represent the relative source per unit time. It is assumed that a linear fit between adjacent time points accurately describes the distribution. As is usual in such specifications of this type, the times may be either ascending or descending, and the source is zero below the lowest valued time and zero above the highest value of time.

Time-Dependent Bin Specifications

If the time dependence of the source is known only by the average value within give time bins, this may be specified to COG by giving:

BIN $t_1\ \bar{s}_1\ \ t_2\ \bar{s}_2\ \ t_3\ \bar{s}_3\ \ldots\ \bar{s}_n\ t_{n+1}$ {IMPORTANCE ...}

where the $\bar{s}$ values are the (relative) average source per unit time within each time bin. The times may be either ascending or descending. When choosing an time from this type of specification, the appropriate time group is first picked, and then a specific time is chosen uniformly between the bin boundaries. Outside of the specified bins, the source is zero.

Steady State

Specification of a steady-state source, one with a constant source strength per unit time over all time, is made by the single input:

STEADY

Note that this then must be used for the time dependence in all increments defining the source. In addition, the problem

normalization and output will then be in terms of source particles per unit time or results per unit time.

Angle Dependence

The angle-dependent specification determines the initial direction of a source particle. We make this statement again since users seem to have some misunderstandings about this. The user should be aware that the position dependence specifies a point in space while the angle dependence is a direction vector in space.

The direction specifications involve inputs that are direction cosines. The following definitions apply:

- u Cosine of the angle between a vector parallel to the +x direction and the specified direction vector.
- v Cosine of the angle between a vector parallel to the +y direction and the specified direction vector.
- w Cosine of the angle between a vector parallel to the +z direction and the specified direction vector.

Needless to say, $u^2 + v^2 + w^2$ will equal one and any of these three quantities may take on only values from minus to plus one.

When we define angular dependence, it is normally easier to measure an angle from some reference direction than from the fixed coordinate system. The first input of any angle-dependent specification is then to give this reference— (u_{ref}, v_{ref}, w_{ref}). The polar angle phi (ϕ) is the angle between this reference and a given direction. All inputs to COG are in terms of the cosine of phi. This is defined as:

$$\mu = \cos \phi = u*u_{ref} + v*v_{ref} + w*w_{ref}$$

where (u, v, w) defines the given direction. In almost all cases, the angular dependent source is a function of only this one quantity. Occasionally there are cases where a full angular definition may be needed. This implies that there is a second reference direction, perpendicular to (u_{ref}, v_{ref}, w_{ref}), from which an azimuthal angle, θ, can be measured. This direction is given by (u_a, v_a,w_a) and the angle is measured as if a right-handed system existed.

There are two special methods of specifying the reference direction. The first involves placing the word NORMAL in the input with no additional data. This method requires that the spatial

Specifying the Fixed Source

definition used with this angular definition is a surface source. The reference direction is set to the outgoing normal to the surface. The second method requires the input word POINT followed by the x, y, and z coordinates of a point in space. The reference direction is then along the line drawn from the point at which each individual source particle originates to the specified point. Physical realities mandate that this case is valid only for an isotropic source used with an importance function.

If the entire reference direction input is omitted, COG will automatically insert the values 0 0 1. This gives a reference direction along the +z-axis.

An angular dependent source specification is then preceded by:

$$\text{DEFINE} \begin{bmatrix}\text{ANGLE}\\ \text{A}\end{bmatrix} = \textit{number} \begin{bmatrix} u_{ref} \;\; v_{ref} \;\; w_{ref} \;\; (u_a \;\; v_a \;\; w_a) \\ \text{NORMAL} \;\; (u_a \;\; v_a \;\; w_a) \\ \text{POINT} \;\; x \;\; y \;\; z \end{bmatrix}$$

where *number* is any positive integer number picked to identify this specific definition. This input is then followed by the specific data needed by one of the six different definitions below. An importance as a function of $\cos(\phi)$ or for specific $(\cos\phi,\theta)$ bins may be specified to be used with the angular inputs. The reference direction for the importance is the same as that given to be used with the distribution.

Fixed Direction

A fixed direction source used with a fixed reference direction always starts headed in the same direction as the reference. This is sometimes referred to as a *beam* source. If a fixed direction is used with the Normal option of defining the reference direction, each particle will start its trajectory along the outward directed normal to the defined surface source. Used with the POINT option, each particle will start headed toward the specified point in space. The following is all that is needed to activate this angular definition:

```
FIXED
```

An importance function is not needed with this option since all particles start in the same direction.

The user should be aware, when using this type of angular dependence, that the point flux estimator does not always include the uncollided contribution to the result. This is dependent upon the type of position dependence used with this source. Always check the printed output to be sure that it is included.

Isotropic Distribution

An isotropic distribution is probably the most commonly used of all the angular distributions. It is specified by:

```
ISOTROPIC  {IMPORTANCE ...}
```

If no importance is given, the reference direction has no effect on the distribution.

Point-wise Distribution in Cosine (Phi)

A given angular distribution as a function of the cosine of the angle from the reference direction can be specified by:

DISTRIBUTION μ_1 s_1 μ_2 s_2 μ_3 s_3 ... s_n {IMPORTANCE...}

where the s values represent the relative source per steradian at the corresponding values of μ The μ values may be in either increasing or decreasing order. Like all the other inputs of this variety, a linear fit is assumed between adjacent points and s is assumed to be zero at all values of μ above or below those provided in the input.

Bin Distribution in Cosine (Phi)

A distribution of the angular source strength known only by average values within a specific bin structure may be specified by:

$\begin{bmatrix}\text{BIN}\\ \text{BINS}\end{bmatrix}$ μ_1 $\bar{s}_1$ μ_2 $\bar{s}_2$ μ_3 $\bar{s}_3$ ... $\bar{s}_n$ μ_{n+1} {IMPORTANCE ...}

The $\bar{s}$ values are the average relative values of the source per steradian within each bin and the bins may be specified as either ascending or descending. Choices within this type of input are

Specifying the Fixed Source

made by first randomly choosing a bin and then picking a value of μ uniformly within the bin boundaries. The source is zero outside of the specified bin structure.

General Bin Specifications in Cosine (Phi) and Theta

For a complete description of an angular source in both the cosine of the polar angle, μ, and in the azimuthal angle, θ, it is possible to provide the following inputs:

GENERAL $min\text{-}\mu_1$ $max\text{-}\mu_1$ $min\text{-}\theta_1$ $max\text{-}\theta_1$ $\bar{s}_1$

$min\text{-}\mu_2$ $max\text{-}\mu_2$ $min\text{-}\theta_2$ $max\text{-}\theta_2$ $\bar{s}_2$

. . . . . .

$min\text{-}\mu_n$ $max\text{-}\mu_n$ $min\text{-}\theta_n$ $max\text{-}\theta_n$ $\bar{s}_n$

{IMPORTANCE i_1 i_2 i_3 ... i_n}

The $\bar{s}$ values are the average relative values of the source per steradian within the specified bins, while the minimum and maximum values define the bin. Theta is input in degrees. Bins must not overlap, but the set of all specified bins need not cover all the existing angular space. Those unspecified regions will be considered to have zero source strength. After a choice of a particular bin is made randomly, the values of μ and θ are picked uniformly within the bin boundaries. Note that the importance is specified by the average values within each bin.

The specification of the reference angular direction for this type of input must always be the specified polar and azimuthal directions.

Needless to say, inputs of this type can easily get difficult to provide just because of the mass of data. To simplify this problem, COG has an established standard bin structure that provides angle bins with equal solid angles. These are used both here in the source angle definitions and also in the detector specifications. Thus, the result of one COG problem can more easily be input to a subsequent COG problem. The input for this form is:

GENERAL N = *number* s_1 s_2 s_3 s_4 ... s_m {IMPORTANCE ...}

Increment Specifications

A single-source increment is specified with the following input statement:

```
[INCREMENT]  source-strength  [POSITION]  =  number-p
[INC      ]                   [P       ]

[ENERGY]  =  number-e    { [TIME]  =  number-t }
[E     ]                  [T   ]

{ [ANGLE]  =  number-a }   { [IMPORTANCE]  =  importance }
  [A    ]                  [I           ]
```

where:

source-strength — The number of source particles born in this increment. This is an absolute value to which the problem results are normalized. If a steady-state problem is being run, the number is the number of source particles per unit time born in this increment.

number-p — Represents the number of the position-dependent source specification to be used in this increment.

number-e — Represents the number of the energy-dependent source specification to be used in this increment. Note that only one type of particle can be specified for use in a single energy increment description.

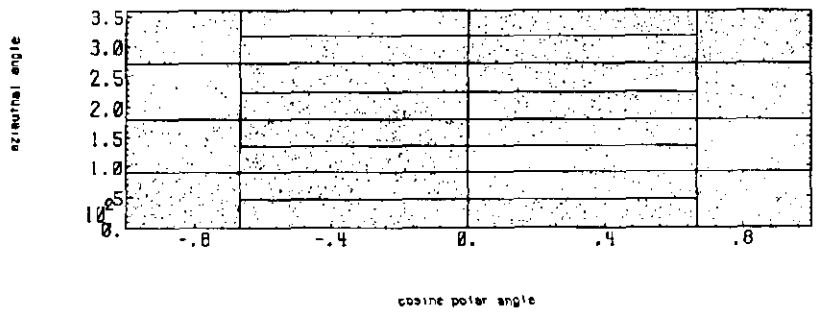
number-t — Number of the time-dependent source specification to be used in this increment. If omitted, the time description reverts to a delta function at t=0.

number-a — Number of the angle-dependent source specification to be used in this increment. If omitted, the source is isotropic.

The value of *number* represents the complexity of the bin structure. There will be $m=4n(n+1)$ bins where n is this input number. The output from a sample problem, shown below, illustrates the structure for n=2 and also shows some of the graphics that accompany this type of input. When you want to run a problem using this feature, we advise that you try a quick run and print out the boundaries for cases that might interest you.

```
SOURCE--ANGLE: Definition of description identified by the number   1

   reference direction for polar angle is given by
        u= 0.          v= 0.          w= 1.0000e+00

   reference direction for azimuthal angle is given by
        u= 1.0000e+00  v= 0.          w= 0.

   input as average values in a series of general bins

   standard bin structure set-up for n=  2
```

cos(phi-min)	cos(phi-max)	theta-min degrees	theta-max degrees	source	importance
1.0000e+00	6.6667e-01	0.	9.0000e+01	4.0000e+00	1.0000e+00
1.0000e+00	6.6667e-01	9.0000e+01	1.8000e+02	2.0000e+00	1.0000e+00
1.0000e+00	6.6667e-01	1.8000e+02	2.7000e+02	4.0000e+00	1.0000e+00
1.0000e+00	6.6667e-01	2.7000e+02	3.6000e+02	1.0000e+01	1.0000e+00
6.6667e-01	0.	0.	4.5000e+01	6.0000e+00	1.0000e+00
6.6667e-01	0.	4.5000e+01	9.0000e+01	4.0000e+00	1.0000e+00
6.6667e-01	0.	9.0000e+01	1.3500e+02	2.0000e+00	1.0000e+00
6.6667e-01	0.	1.3500e+02	1.8000e+02	4.0000e+00	1.0000e+00
6.6667e-01	0.	1.8000e+02	2.2500e+02	6.0000e+00	1.0000e+00
6.6667e-01	0.	2.2500e+02	2.7000e+02	8.0000e+00	1.0000e+00
6.6667e-01	0.	2.7000e+02	3.1500e+02	1.0000e+01	1.0000e+00
6.6667e-01	0.	3.1500e+02	3.6000e+02	9.0000e+00	1.0000e+00
0.	-6.6667e-01	0.	4.5000e+01	4.0000e+00	1.0000e+00
0.	-6.6667e-01	4.5000e+01	9.0000e+01	6.0000e+00	1.0000e+00
0.	-6.6667e-01	9.0000e+01	1.3500e+02	8.0000e+00	1.0000e+00
0.	-6.6667e-01	1.3500e+02	1.8000e+02	1.0000e+01	1.0000e+00
0.	-6.6667e-01	1.8000e+02	2.2500e+02	9.0000e+00	1.0000e+00
0.	-6.6667e-01	2.2500e+02	2.7000e+02	6.0000e+00	1.0000e+00
0.	-6.6667e-01	2.7000e+02	3.1500e+02	4.0000e+00	1.0000e+00
0.	-6.6667e-01	3.1500e+02	3.6000e+02	2.0000e+00	1.0000e+00
-6.6667e-01	-1.0000e+00	0.	9.0000e+01	1.0000e+01	1.0000e+00
-6.6667e-01	-1.0000e+00	9.0000e+01	1.8000e+02	4.0000e+00	1.0000e+00
-6.6667e-01	-1.0000e+00	1.8000e+02	2.7000e+02	2.0000e+00	1.0000e+00
-6.6667e-01	-1.0000e+00	2.7000e+02	3.6000e+02	4.0000e+00	1.0000e+00



importance Relative importance of particles born in this increment compared to the importance in other increments. If omitted, the importance is set to 1.

As many increments as needed to describe the source should be given. Each has the same format as above including the word INCREMENT. When listed in the code printout, an increment number is indicated. These are assigned internally in consecutive order and are for the user's reference only. They are not input.

The following illustrates the code printout of increment data for a steady-state problem. Note that the total source strength by particle type is summed for the user's convenience in checking the normalization of the problem. Time or angle definitions represented by 0 (zero) signify the default values.

```
DEFINITION OF FIXED SOURCE

  total number of source particles to be generated          1000

     increment  source strength    description of separable parts     relative
                                  position  energy   time   angle   importance

         1      3.7000e+10  n/s      1        1       1      0     1.0000e+00

         2      1.8500e+11  p/s      1        2       1      0     2.0000e 01

         3      2.5000e+09  n/s      2        1       1      0     1.4800e+01

         4      1.2500e+10  p/s      1        2       1      0     2.9600e+00


        totals

               3.9500e+10  n/s
               1.9750e+11  p/s
```

Importance values specified in the energy, time, and angle definitions are used to bias the choice of these quantities from their respective distributions. The importance value assigned to increments are likewise used to bias the choice of which increment will be chosen to provide the source description. There are two general cases where one would use increment biasing. The first case occurs if the physical source is spread over a rather large volume but only particles born in a small part of this volume

Specifying the Fixed Source

contribute much to the desired result. The user might then wish to break the volume source into a number of increments, each of which represent just part of the total volume. Those increments contributing most of the result would then have large importance values while those contributing little would have smaller importance values. This effectively allows importance sampling of the spatial distribution of the source. The second case involves a problem with both neutrons and photons. Generally there is a large number of source photons compared to the number of neutrons. Importance values assigned to the increments can alter the selection process, so approximately the same number of source neutrons as source photons are followed. In both cases, of course, the initial statistical weights are altered by COG so that the problem solutions remain correct.

With the definition of all increments and their supporting position, energy, time, and direction definitions, COG is ready to form the initial source parameters for the required number of source particles. These data, when generated, are written onto one or more disk files and will then be used when running the random-walk trajectories.

As part of the formulation process, the code keeps track of the maximum, minimum, and average parameter values of each particle. These values are printed so the user can check to insure that the specified source is correct. As an example, assume that the desired problem represents an isotropic fission neutron source located in a 2-cm radius sphere located at (4,–7,3) and with all particles born in time in a Gaussian distribution at 3.0E-08 second with a full width at half maximum of 1.0E-08 second. No importance biasing was used. The following averages are indicated by the code:

```
average source parameters

                          maximum          minimum          average

   neutron
                  x     5.995e+00        2.014e+00        3.991e+00  cm
                  y    -5.005e+00       -8.979e+00       -7.001e+00  cm
                  z     4.982e+00        1.014e+00        2.993e+00  cm
                  u     9.997e-01       -9.999e-01       -4.104e-03
                  v     9.999e-01       -1.000e+00        1.100e-02
                  w     1.000e+00       -9.997e-01        1.392e-03
                  t     4.538e-08        1.305e-08        3.007e-08  sec
                  E     1.419e+01        1.402e+01        1.410e+01  mev
                 wt     1.000e+00        1.000e+00        1.000e+00
```

The position averages are close to (4,–7,3), as they should be, and no values are outside of the ranges that would be expected ($2<x<6$,$-9<y<-5$, $1<z<5$). The direction cosines each vary from +1 to –1, with an average of zero. This is what would be expected for an isotropic distribution. The time average value is about 3.0E-08 and varies between 1.0E-08 and 5.0E-08. Again, as expected. The energy, however, has an average of 14.1 MeV, a little high for a fission distribution. Obviously, the problem has an error in its input distribution of energy that should be corrected.

User-Created Source File

It is expected that the source routines built into COG will satisfy the needs of almost all users. There will be a very few, however, who need to simulate a source that can not be easily represented by the separable functions just described. A quick guess would indicate that these problems will fall into two categories—those with strongly coupled energy-angle-time-position relationships and those taking the source properties directly from the histories of a previously run Monte Carlo calculation.

Running COG in these cases is not difficult. The trick is to write a source file containing all the required starting data for each particle that will be followed in the random-walk process. This data would be generated by the user using the specific relationships applicable to a fixed physical situation or by saving individual particle data from a previously run problem. COG is already designed to read existing source files without having to generate them.

To write such files, one only needs to know the formats and data storage requirements of COG. Since these are a little messy because of requirements imposed by the point flux estimation techniques, they will not be given in this manual but reserved for a subsequent volume that will describe the fine details of the source program. If a user has a need for such information before publication of that volume, he should contact one of the authors for assistance.

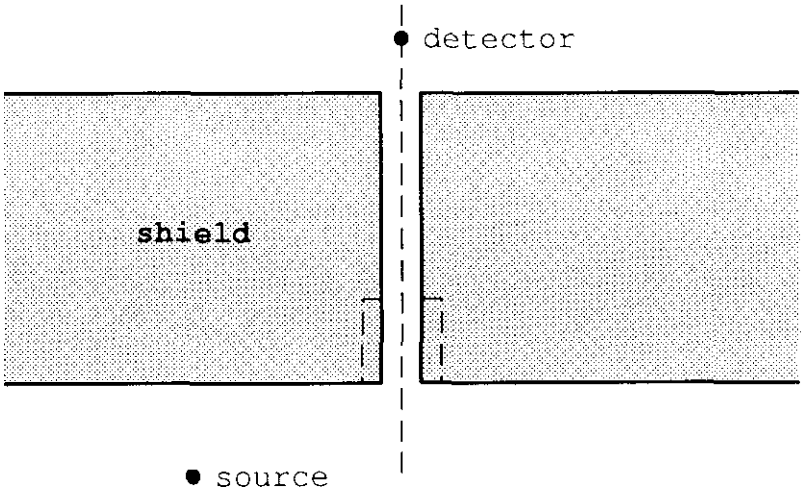
Walk-Source Option

A few years ago we implemented a special Monte Carlo method to obtain results for a particular class of problems. Our success with this method caused us to add it to COG as a standard source option. The method involves calculating the uncollided particle flux from a defined source at randomly chosen points within a specified

Specifying the Fixed Source

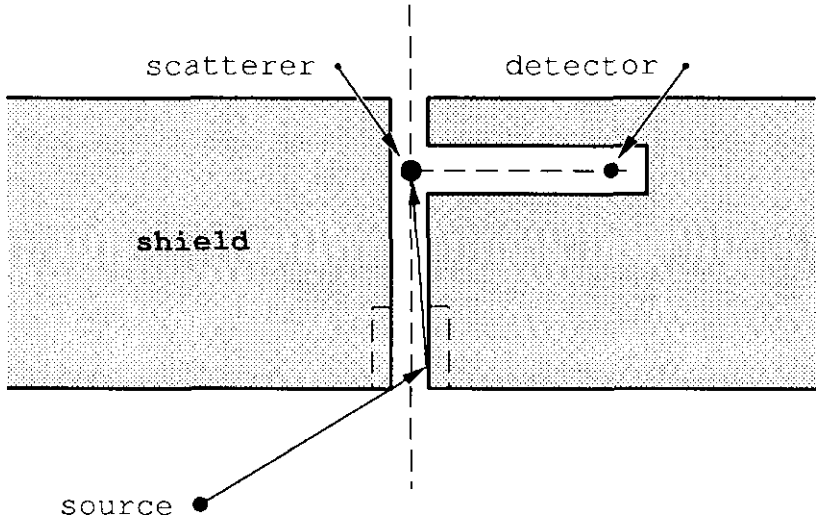
volume. This is then used to force collisions at these points and the exiting particles form the starting source for the random walk.

The following is a simple illustration of a problem where this method could be used advantageously. Given a slab shield with a small cylindrical hole running through it, place the detector at the back side of the hole and a source on the front side but not in line with the cylindrical hole.



The first collision scatterings in the small cylindrical annulus around the hole on the source side of the shield produces particles that contribute almost all the response of the detector.

This method has been expanded to consider not only forced first collision sources but also forced second and third collision sources. The second collision-forced source could be illustrated by adding to the above example another cylindrical hole at right angles to the first and adding a scattering material located within the first hole adjacent to the second.



Similar examples for the third scattering collision can be imagined.

In a complex geometry, it is normally easy to identify several paths that could be followed by source particles to reach the detector position. Each of these paths may have zero, one, two, or three collisions and each must be specified in the code input. The physical source must also be specified. Omission of a meaningful path implies that the calculated result will be lower than the actual result obtained in nature. If there are too many paths that must be defined, this method probably should not have been used.

There are two ways of looking at this method. The first is that it is a modification of the random walk and the second is that it is just another way of representing the source. Both are correct. Since we traditionally have considered it at LLNL as an alternate source description, we have added it to the source descriptions of COG and discuss it in this chapter. At the same time, we have given it a block designation-name of a random-walk modification.

The user should note that if two or more paths are defined that are identical to each other or that contain overlapping pieces that are identical, the calculated results will include contributions from each and will yield results that are too high.

Specifying the Fixed Source

To activate this method, insert a block of input data with the name WALK-SOURCE, and then include the following data:

```
PATH number (V1 = number-1) (V2 = number-2) (V3 = number-3)
                {I = importance}

PATH number .....................

PATH number .....................

.................................

DEFINE VOLUME number-a  [ BOX        ..... ]
                        [ CYLINDER   ..... ]
                        [ CONE       ..... ]
                        [ SPHERE     ..... ]

DEFINE VOLUME number-b ..........
DEFINE VOLUME number-c ..........
.................................
```

where:

number — An integer assigned by the user to a given path. It is used only to keep track of the paths in the user's mind.

number-1 — A user-assigned positive integer associated with the volume in which the first collision of the specified path will occur. If omitted (along with V2 and V3), the source will be started normally as specified by the SOURCE block of data.

number-2 — A user-assigned positive integer associated with the volume in which the second collision of the specified path will occur. If omitted (along with V3), the source will be started in volume-1.

number-3 — A user-assigned positive integer associated with the volume in which the third collision of the specified path will occur. If omitted, the source will be started in volume-2. If included,the source will be started in volume-3.

importance The relative importance of this path related to the other defined paths. If omitted, *importance* will be set to one.

number-a One of the numbers used to define a volume in the path definitions. The entire set of DEFINE VOLUME inputs must define all the numbers used in the PATH statements.

The inputs to specify the volumes are identical to those given for volume sources in the section **Position Dependence** of this chapter (page 113). Alternate names, such as SPH and S for the sphere, are allowed as well as the restriction to a given material or sector number.

Walk-Channel Option

The previously described WALK-SOURCE option is useful for many cases. However, if a particle makes a great many collisions before entering into the regions of interest, this option is of no value in obtaining a solution. The following is an example of such a situation.

There is a neutron source encased in a thick iron container. The hydrogenous material normally present has disappeared in a postulated safety situation. The container sits in a concrete-shielded room, and, for all practical purposes, the only opening in the shield is a small hole leading to an adjacent room. It has already been determined that the concrete is thick enough to protect adequately those in this second room from particles impinging directly upon the wall. Therefore, it is desired to calculate the dose in the second room from particles going through the hole. The physics of the scattering in iron dictates that the neutrons leaking from the iron container will have scattered many times. The particles that then leak have a very small probability of ever hitting the hole.

On method of obtaining an answer to this problem is to run two separate computer problems. The first would use point-flux techniques to obtain a particle current at the source end of the hole. The second would use this current as a starting source to obtain the desired dose in the second room. We have combined both these problems into one COG run by allowing the user to specify the WALK-CHANNEL option.

Specifying the Fixed Source

The inputs to this option provide the following information to the code:

- A definition of the surface on which the coupling current is to be calculated.
- The regions in which source and scattering events are to be used to determine the coupling current.
- The region in which the surface is located.

Particles are generated by the usual source definition and allowed to follow a normal or biased random walk. Source events and collisions in the specified regions are used to calculate the current at a randomly chosen point on the defined surface. This current is stored in a bank and will later be used to follow a continuation of the particle path beyond the surface. A particle crossing into the region that includes the surface from one of the regions used to calculate the current will be Russian-rouletted out of existence. If it were not, the problem would contain twice the correct current—that indicated by the random walk and that predicted by the point estimation. A particle generated by this method is not allowed again to produce a contribution to the calculated current.

To activate this option, input the block name `WALK-CHANNEL` followed by the definition of the coupling surface from one of the following:

```
SS-TRIANGLE ..................
SS-PARALLELOGRAM .............
SS-DISK ......................
SS-SPHERE ....................
SS-CONE ......................
SS-CYLINDER ..................
```

where the complete input parameters are specified on pages 115-118 of this manual and are identical to the spatial definition of a fixed surface source.

This is followed by:

REGION n_1 n_2 n_3 ..... n_i

where the n_i represents a list of the region numbers in which source and scattering events will be used to calculate the current.

Finally, the region number *m*, in which all the surface defined above occurs, is specified with the addition of :

```
TERMINATE m
```

Results are normalized to the COG-defined source, as in all other problems.

Specifying the Fixed Source

The Random Walk

Explanation of Random Walk Events

Analog Monte Carlo

In a normal unmodified random walk, the position, energy, type of particle, initial direction, and time of starting are picked from the given source distribution. The number of mean-free paths the particle will travel is then chosen randomly from an exponential distribution. The position of the first collision is determined using the geometric description, mixture compositions, and cross sections provided with the problem. The cross-section information of the material at the collision site then allows a random selection of the particular isotope involved in the collision as well as the particular type of reaction. Physics data then allow the selection of one, or more, outgoing particles. The entire process is repeated again and again for each of these particles until all source, scattered, and secondary particles have either been absorbed or have leaked from the defined problem.[19]

An *answer* to a specific problem is normally obtained by looking at the random walk events in some relatively small geometric volume and throwing away those that did not occur in a given time interval. The random walk events remaining are weighted by some function of the energy and direction of travel of each individual particle, and then the weighted events are summed and divided by the total number of source particles. Thus, only a few events in phase-space (position, energy, time, and direction) are *scored* to obtain the answer.

If the position is relatively *close* to the source, there will be a fairly large number of scoring events occurring per source particle that can be used to obtain the answer. For a *deep-penetration* problem, the number of scoring events occurring per source particle is small.

Now, in very general terms, the statistical uncertainty associated with an answer is proportional to the inverse of the square root of the number of events used in computing that answer. Thus if we have an answer with a 10 percent fractional standard deviation, we would have to run that problem 100 times longer to obtain an answer with a 1 percent fractional standard deviation. If the

original problem ran in 10 seconds, the more accurate problem would run in 1000 seconds, or about 17 minutes. For those of us who have very low dollar costs for machine time, this would be a very acceptable problem run. For a deep-penetration problem, we might well calculate an answer with a 50 percent fractional standard deviation after 17 minutes. To achieve the 1 percent result we would then have to run for more than 29 days. At that point we become interested in ways to obtain our answer with less running time. Thus, the interest in the contents of this chapter.

Before we get too involved in describing ways to run faster, let's get one very important point across. The analog calculation, as described in the opening paragraphs above, is the safe way of doing the calculation. If you can get an acceptable result without making modifications to the random walk, by all means do so even if you use a lot of computer time. Any modification to the random walk increases the risk of obtaining an incorrect answer.

Biased Monte Carlo

The general idea in modifying the random walk is not to waste time in following particles that will not contribute to the *score*. There are two general ways to implement this concept. The first way would be to terminate the history of any particle when you determine that it would not be of any further interest. An example of this could be a problem in which the result is desired only for times less than 10 microseconds. In any random walk when the age of a particle exceeds 10 microseconds, the particle could be dropped from consideration and not followed.

The second way would be to enhance the effort expended on a particle in the right place and/or headed in the right direction. Thus, some methods *split* such particles into two separate particles and follow each independently to sample from the possible random walks more frequently.

To achieve a quick, fast, and accurate problem solution, one would like a machine code to implement one, or more, of the random walk techniques automatically during a problem run. There have been attempts to do this, and some codes have been written with various methods built into them. We did this about 14 years ago in MORSE-L[9]; the Mathematics Application Group, Inc. (MAGI) spent some time developing concepts[20]; and LANL did some recent work in the field.[21] Most of the built-in or automatic techniques were found to work well only with a fairly fixed type of problem—they

actually made other types of problems run more poorly. COG does not include automatic techniques; however, it does produce output that enables the user to look at a short problem run, determine what is happening, and then put the information he decides will speed up the calculation into a subsequent problem. In the next generation of Monte Carlo codes it may be possible to build in some of these techniques directly and bypass the user.

It helps to understand one concept when working with random walk modification. This is the idea of *statistical* weight. If you have a particle in the random walk and decide to split it into two particles, obviously each of these can only contribute half as much to an answer as the original particle would. To do this we assign to a particle a statistical weight. Starting with an unbiased source, this is set equal to one. If this particle is split, each of the new particles would weigh one half. If each of these two particles undergoes splitting, the resulting four particles would each have a statistical weight of one quarter.

The score then uses the statistical weight in its calculation. In cases where the natural distribution used to choose a direction or an energy is biased, the statistical weight represents the proper proportion to use when obtaining the answer and is not a simple number like one half. The user never sees the statistical weight—it works behind the scenes to insure that the correct result is obtained. Thus two identical problems, one run in the analog mode and one run with random walk modifications, will produce the same result if both are run long enough. The only difference would be a (hopefully) smaller statistical error in the problem run with random walk modifications.

Another idea should be strongly emphasized. All changes to the random walk will tend to force particles toward some preferred point in *phase space*. If done correctly, this is the point at which a detector is located or over which detector contributions are calculated. In general, this is only one point and only one detector is located there. Therefore, a problem with a given set of random walk modifications will produce better results at only this one detector position and will produce poorer results at all other detector positions. If you try to pick random walk modifications to give better results at many detector positions, you will end up finding that the proper modifications are no modifications—in other words, an analog problem run.

If you have several detectors close to each other, you may very well find that one set of modifications produce acceptable results for all

detectors. But, as the detectors move apart from each other, you will do better to run one problem for each detector. Each of these problems, of course, would have modifications matched to the individual detector location. Theory, according to work done at ORNL[22], indicates that there is an optimum set of modifications that will produce the exact result with only one source particle. However, it takes more effort to find this set of modifications than it takes to solve the analog problem. Most of us are willing to accept a *good* guess for the modifications.

COG allows the user to employ many different random walk modifications. All are functions of statistical regions, particle types, and particle energies. Some are simple techniques that terminate the random walk when a particle has an *age* greater than some input time, or stops secondary particle generation, or forces particle survival when calculating the events occurring at a collision site. A series of more complex modifications cause splitting or Russian roulette to occur at collision sites or at boundary crossings. One does a path stretching toward, or away from, a preferred point, sphere, cylinder, ring, or direction. Others modify the scattering process to bias the scattered energy or direction. Another forces collisions to occur in given regions. Not to be forgotten are those choices that bias the initial source generation and modify the point flux estimators. We added almost everything that others have tried and a few new items. It is to be expected that additional methods will be added to this list in the future.

In general, a user will make use of only one or two of these modification techniques in the same problem. We allow a great range of choice, because users have their favorite methods. While each of the options have times they should be used and times when they shouldn't, the choice is often a matter of personal likes and dislikes. Thus, splitting and path stretching accomplish the same thing and are mathematically the same. In some cases it is easier to specify path stretching and in others it is easier to specify splitting. Both can get you into a lot of trouble, and both can do wonders in securing a good result. In the following section we have tried to indicate something about each of the modification techniques, but our knowledge is limited to only our experience. The comments given are therefore not inclusive of all knowledge.

Importance

Many inputs use the term IMPORTANCE. In the chapter **Specifying the Fixed Source** (page 109), this term was used, but it was not defined in detail. We should do this now, since the concept is basic to random walk modification. Importance is the ratio of a *result* obtained at a specified detector to a unit source at some place in phase space—that is,

$$\text{importance}(\vec{r}, E, t, \vec{\Omega}) = \text{result}(E', t', \vec{\Omega}') \;/\; \text{source}(\vec{r}, E, t, \vec{\Omega})$$

Places in phase space with high values of importance contribute a lot to the result. Places with low importance values contribute only a little. Note that the result is automatically a function of position, even though this quantity was not given in the above definition.

As you work with importances you will come to understand a very horrifying fact—importances are a function of a lot of variables if the desired results have a lot of variables. As an example, let's take the importance associated with a fixed source. To make things easier, let this fixed source be at a point in position, with a fixed initial direction, and at a delta in time. So all we have left to bias is energy. For a result that is integrated over all energies, times, and directions, we then have:

$$\text{importance}(E) = \text{result} \;/\; \text{source}(E)$$

What this means in terms of a problem is: given a source at energy E and the result from this source, the importance is the ratio of the two. In the real problem, weighting the fixed source by the importance should give a better problem result than not weighting the source. Now, say that we really wanted a problem result that was time dependent. we would then have:

$$\text{importance}(E) = \text{result}(t') \;/\; \text{source}(E)$$

Now this says that there are a set of importances as a function of source energy for each time at which results are desired. If you want a good calculation at an *early* time you will use a different set of importances than those you would use for a *late* time.

COG will calculate importance values for you. The details are given in the detector definitions presented in the chapter **Specific Results Using Detectors** (page 167).

The Random Walk

As you try different importances and different schemes for modifying the random walk, you will want to know how successful you have been. The standard measure of this is the *figure-of-merit* defined as:

$$FOM = 1.0 / (FSD^2 \times (running\text{-}time))$$

where FSD is the fractional standard deviation—one standard deviation divided by the result. The *running-time* is the time spent in the random walk calculation.

There are problems with this quantity, so do not assume it has great accuracy. For example, it means nothing between two unlike problems because the geometry and physics of particles slowing down are different. Be cautious that problems you compare are both similar and have the same detectors or the running time, which includes the time to perform detector calculations, will not be compatible. The idea is to compare the figure-of-merits for two similar problems with different approaches to the random walk modifications. The problem with the larger FOM has run more effectively.

User Modifications to the Random Walk

The following sections present data that activate various methods of modifying the random walk. Each represents a block of input, preceded by a block name, and may be specified only once in any one problem. If any block is omitted, the specified modifications are not made and the random walk continues normally without the indicated changes. If all the indicated blocks are omitted, the problem will be run in the analog mode except for any specifications of importance given in the source or the activation of the WALK-SOURCE or WALK-CHANNEL options.

All the inputs defined below are functions of statistical regions, particle types, and particle energies. Since each type of input has identical input formats for these quantities, we will describe them only once. For each particle type there exists a two-dimensional grid that lists statistical region number against particle energy. The inputs provide data for specified areas on these grids. It is implied that unspecified areas do not use the input option. The first part of the input is the particle type. The particle type is followed by the region to be covered and the various energy ranges to be used within this region—each energy range is followed by its required data. The next region is then specified followed by its

energy ranges and data. After all regions have been given, the next particle type with its associated inputs is specified.

The region part of the specifications are given as:

```
[REGION]     [ALL or A      ]
[REG   ]     [n1 TO n2      ]
[R     ]     [n             ]
```

Where *n* represent a statistical region number. The input ALL specifies that all regions are covered. The specification n_1 TO n_2 means that all regions equal to or greater than n_1 and equal to or less than n_2 are given the appropriate definitions. A single number means that the description of only one region is provided.

The energy part of the inputs has the following form:

```
[ENERGY]
[ENG   ]   energy TO energy
[E     ]
```

The input *energy* represents a specific value of energy in the units being used in the problem. An example of an input would be:

```
WALK-PS

NEUTRON
          REGION  1 TO 7    ENERGY 0.0 TO 0.5    [.....]
                             ENERGY 0.5 TO 1.0    [.....]
                             ENERGY 1.0 TO 20.    [.....]
          REGION  8 TO 10    ENERGY   0 TO 1.0    [.....]
                             ENERGY 1.0 TO 10.    [.....]
PHOTON
          REGION  1 TO 3     ENERGY   0 TO 20.    [.....]
          REGION  4          ENERGY   0 TO 20.    [.....]
          REGION  5 TO 10    ENERGY   0 TO 1.0    [.....]
                             ENERGY 1.0 TO 20.    [.....]
```

Note that each particle type occurs only once in a specific WALK-*XX* block of data. The above example is reproduced exactly as input—

there are no ditto marks left out.. The last physically specified input overrides all previous inputs.

Particle Termination Based On Age

This option terminates a random walk when the particle has reached a specified age upon exit from a collision. Age is the time associated with the particle. It is equal to the initial starting time from the source description plus the time spent traveling to the current position of the particle. Normally, this termination is used when the desired results are to be calculated before a set time value. The input, then, would be this time value or, more correctly, this value less the transit time from a statistical region to the detector point. Remember that slow neutrons can produce fast photons.

The input block name for this option is WALK-AGE, and the number required for this option is a single value equal to the cutoff time.

An age termination is very straightforward and normally produces no bad effects. An incorrect input larger than the desired, or proper, value just makes the problem run longer. A smaller value results in a smaller than correct result.

Secondary Production Control

The input block name for this option is WALK-PRODUCTION and no input data are required. This implies that the definition of the area in the region–energy specification is sufficient to indicate that the option is wanted. The option, in this case, suppresses all secondary particle production in reaction events. The specified particle type in the input indicates the incident particle type in this control.

A user would wish to activate this option in parts of a problem if he recognized that secondary particles from those parts would never contribute meaningful amounts to the detector result. No adverse effects would occur unless the suppressed particles really would make a meaningful contribution.

Survival of a Particle Undergoing a Collision

When a particle undergoes a collision, there is a probability that it will be absorbed without producing secondaries. This terminates the random walk. There are times when a lot of calculational

effort has taken place to get the particle to this point in the problem, and it would be desirable to keep it alive to get more information from this random walk. This option, when used, does not allow the particle to be absorbed but just decreases its statistical weight to represent the probability of nonabsorption.

The block name for this option is WALK-SURVIVAL, and no additional input data are required as explained above.

There are two problems that have been encountered when using the survival method. The first is that you can get a lot of scores in your answer from only a few, or even one, random walk histories. This implies that you are kidding yourselves about the validity of the calculated results. There are many paths in a real problem in moving a particle from point A to point B. If you only have one particle that makes it to B, and then you bounce it around a lot and generate a lot of individual scores in getting an answer, you still have not sampled enough from the paths from A to B to have a good answer. COG tends to point this out to you by determining only one score per source particle—the sum of all these contributions. Still be very careful in how many source particles contribute to the total result.

The second problem does not cause calculational errors but does cause headaches. If you never allow a particle to die because of absorption, and you do not put in some other method of stopping it, you will never finish with the first source particle. Always mix the survival method with one of the other options to kill off undesired low weight particles. Weight windows are excellent for this purpose.

Forced Collisions

In some types of problems, you do a lot of work to get a particle to a given region, say a thin foil, and then the particle goes right on through the region without having a collision. If a successful problem result depends on having collisions, you have to run very large number of source particles before there are a meaningful number of collisions and an acceptable result. This option provides a way of solving this type of problem by forcing collisions to occur.

When a particle goes through a region activated by the appropriate inputs, it will be split into two particles. One will go through the region without interaction, while the other will be forced to have a collision. The particle forced to have a collision will not be allowed

to have another forced collision during the remainder of its random walk. The other particle may undergo another forced collision sequence if it encounters a so-designated region.

The required block for this option is labeled WALK-FC, and there are no additional data required beyond the usual particle type, region, and energy specifications.

Just like the survival technique, this method can give you a lot of random walk events based upon just a few histories. Be aware of how many events are taking place and whether you have created too many forced collision events. It is also possible to be forcing collisions in a thick region. In this case, the exit particles represent uncollided particles, and the forced collisions will be toward the entry point of the region. Both of these can yield incorrect histories. The method was designed for *thin* regions—those less than about 0.5 mean-free-paths in thickness. If the region is thicker than this, you do not need this method.

Splitting and Russian Roulette at a Collision Site

Splitting and Russian roulette are the two most commonly used techniques to modify the random walk process. They are also the most commonly misused techniques. Normally, they are treated as separate modifications but within COG we have combined them, so a problem can use only one at a given position in space and energy. This stops users from both splitting and killing off particles at the same time. The code does offer three different ways to implement these techniques. This section discusses their use at a collision site.

Activation of this method is initiated by including the WALK-COLLISION block of input information. The information needed for each particle type, statistical region, and energy is the quantity β—defined as the desired number of particles, on the average, exiting a collision divided by the number entering the collision. If β is set to zero, there will be no exiting particles. If β is 0.5, statistically one half of all exiting particles will be killed and the remaining half, which will have their statistical weight doubled, will continue to be followed. If β is 1.0, a normal random walk will result with no Russian roulette or splitting. For a β of 1.5, statistically one half of the exiting particles are unmodified and the remainder are split into two particles, each with half the statistical weight of the original particle. For a β of 2.5, statistically one half of the exiting particles are split into two particles, while the remainder are split into three particles.

Values of β may take on any positive value and are not limited to those examples given above. Normally, β has values greater than 1.0 at high particle energies and less than 1.0 at low energies—i.e., one splits at high energies and plays Russian roulette at low energies. One would expect that where the importance is high, β should be greater than 1.0; and where the importance is low, β should be less than 1.0.

The problem with splitting is that you may oversplit. One source particle can be divided so often that there are thousands of particles being scored at the detector. It is easy to forget that the source has not been sampled sufficiently enough to have a good result or that adequate initial paths have not been followed. Again, COG attempts to indicate this by summing all detector results over a single source particle before statistical parameters are calculated.

Splitting and Russian roulette occur independently of the statistical weight of the particle. Somewhere there is the idea that these weights are Gaussian distributed. They certainly are not in an analog problem; and as a function of just position and energy, one would not necessarily expect that they would be. In fact, at any spot in position and energy, it is not difficult to devise a problem where particles of very low weight contribute almost all of the score to a time-dependent answer, while the higher weight particles contribute little. Thus, COG does not get into trouble with this form of splitting by considering the individual particle weight.

One of the nice features of this particular method of splitting is that the relative distance between splitting events is determined by the cross-section properties of the material in which they are traveling. It is not necessary to establish boundaries one mean-free-path apart; the technique itself spaces the splitting events at collision sites. This is also true of the Russian roulette events.

Splitting and Russian Roulette at a Boundary Crossing

Splitting or Russian roulette is attempted every time a particle crosses a boundary into a different numbered statistical region when this method is used. The input data are in the WALK-BC block of information and include, for each statistical region and energy, a value of importance. The value of β, as described above, is evaluated by the ratio of the importance in the region being entered divided by the value of importance in the region just exited. Importances not specified in the input are automatically set to one.

The Random Walk

All the comments made about over-splitting in the previous section apply here. H. Goldstein used to set a criterion for this type of random walk modification. For a problem with a set of boundaries for splitting that were between the source and the detector, the criterion stated that approximately the same number of random walk particles should cross each. This is not a bad criterion to use.

The use of boundaries to determine where things are to be modified gives the user a lot of control. It also requires more effort when initially defining the problem. Again, COG does not consider the statistical weights when deciding to split or kill. This eliminates some pitfalls but may not achieve some desired effects.

Splitting and Russian Roulette at a Collision Site Using Statistical Weights

In this version of splitting and Russian roulette, the particle exiting from a collision is checked. If the particle statistical weight is greater than a given maximum weight standard, splitting is done using a method that gives all the resulting particles a weight equal to the maximum weight standard. If the exiting particle weight is less than a given minimum weight standard, Russian roulette is performed with any surviving particle having a weight equal to the minimum standard. This is a technique used in the ORNL code O5R[6] and in the LASL code MCNP[5] where it is called by the name *weight window*.

As long as you know that high-weight particles are high-importance particles and that low-weight particles are low-importance particles, this method offers some nice features. It is difficult for the user to determine the proper input values for maximum and minimum statistical weights. It is of little value to say that they are related to the values of particle flux, because if you knew the flux, you would know the answer and wouldn't run the problem. LASL has developed some techniques to output these quantities, and when time permits we will add similar outputs to COG.

This method is activated by including the WALK-WT block of information. The required data for each particle, region, energy combination are the minimum weight standard followed by the maximum weight standard.

Path Stretching

Path stretching is a technique that involves biasing the choice of the distance a particle will travel. This distance is chosen from an exponential distribution. The bias involves specifying a preferred direction and then a parameter that specifies how much further the particle will be allowed to travel if it is going toward the preferred direction. If it is going away from the preferred direction, the particle's path will be shortened.

The technique of path stretching is sometimes referred to as an exponential transform.[23] As in any transform, the basic equation (in this case the Boltzmann transport equation) has a substitution made in it for the independent variable. After some algebraic manipulation, a new equation is obtained. The solution of this equation, which is supposed to be easier than the original equation, results in a value of the new independent variable, and this is converted back to the original variable. The particular substitutions normally employed in the exponential transform method leads to a solution of the Boltzmann equation with path stretching. However, the solution also involves a biasing of the fixed source. Because in COG the biasing of the fixed source is left to the user, we will use the term path stretching rather than the term exponential transform.

When path stretching is employed, the effective number of mean-free paths a particle will move is altered by the factor:

$$1.0 - q\mu$$

where q is an input number dependent upon particle type, statistical region, and energy. Values between –1 and +1 are allowed. The quantity μ is the cosine of the angle between the direction of travel and the preferred direction. Values of q less than zero will move particles away from the preferred direction. A value of q equal to zero implies that the method is not used, while values greater than zero move the particle toward the preferred direction.

Obviously, the problem with this technique is how to determine the proper value of q to use in any specific problem. A single value used throughout all of phase space rarely provides a good solution. To understand the choice of q, one should understand the physics of particle movement through materials.

The Random Walk

If we have a source at region A and a detector at region B and some material between these, then the *result* at region B depends upon how particles move from A to B. If the particles are neutrons and the source has a high-energy distribution, the highest energy neutrons will move to B with few or no collisions and then have collisions giving a neutron energy distribution. Lower energy neutrons born at the source will never make it to A. This is the old *two-group* idea used for hand calculations. A large value of q, which moves the high-energy particles to B with few collisions, will then work quite well. If, however, the initial source is made up of low-energy neutrons, the transport from A to B will be accomplished by a long series of scattering collisions. Path stretching then must insure that these collisions occur and can, at best, just make sure that we follow particles headed in the correct direction.

The good points of this technique is that it does not require a lot of extra boundaries, that it is employed at each collision site, and that it is sensitive to particle direction. The problems are related to the proper choice of q.

Path stretching is activated by including the WALK-PS block of data. Data required for each particle, region, and energy specification include the value of q followed by the definition of the reference direction. This definition is chosen from the following:

Definition	Explanation
POINT $x\ y\ z$	Towards a given point in space as defined by the location x, y, z.
[SPHERE / SPH / S] $r\ x\ y\ z$	Towards the closest point on a spherical shell with its center at x, y, z and with a radius r.
[CYLINDER / CYL / C] $r\ x_1\ y_1\ z_1\ x_2\ y_2\ z_2$	Towards the closest point on a cylinder defined by two points on its axis and with a radius of r.

[RING / R] $r\ x_0\ y_0\ z_0\ x_1\ y_1\ z_1$	Towards the closest point on a ring with its center at point 0 and with point 1 on its axis and with its radius equal to r.
[DIRECTION / DIR / D] $u\ v\ w$	Preferred direction specified by direction cosines u, v, w.

Scattered Energy Bias

As a particle undergoes a collision, normally, some distribution of emerging particle energies can result. A normal random walk chooses from these based upon natural probabilities of occurrence. The scattered energy bias option modifies these choices based upon user-specified sets of importance functions. The option is chosen when the WALK-SEB block is included. The information associated with each particle, energy, and region specification is the appropriate importance distribution:

$$e_1\ i_1\ e_2\ i_2\ e_3\ i_3\ \ldots\ldots\ e_n\ i_n$$

where the e values are either ascending or descending. As usual, the importances are zero outside the range of input values.

The scattered energy bias method is useful to prevent making a lot of low-energy particles in cases where these particles will not travel to the regions of interest. In many cases, however, there is not a lot of choice in the exit particle energies, and scattered energy bias does not accomplish much.

Scattered Direction Bias

A problem occasionally occurs in which you wish to modify the scattering so that more particles will scatter in a preferred direction. The scattered direction bias option allows particles to be guided around corners. Within COG this is made possible by inserting the WALK-SDB block of inputs. For each particle, region, and energy, the inputs contain the strength function b and the reference-direction. The reference direction is specified in the same manner as the path stretching method illustrated in the section **Path Stretching** (page 157).

The relative importance assigned to each direction is determined by the strength function, b. It is defined as:

$$\text{importance} = 1 / (1 + b \cdot (1 - \mu))$$

where μ is the cosine of the angle between the preferred direction and the exit direction of the particle. Note that this is a function of only one angle and that the importance relationship is automatically defined. It is expected that there will be problems enough in finding the correct value of b with these simplifying assumptions. An importance based on a full three-dimension angular representation would be too difficult ever to express with any degree of accuracy. An input value of b equal to zero results in equal importance in all directions while increasingly larger positive values of b result in more peaked importance toward the preferred direction.

Code Output Illustrating the Random Walk Results

As stated many times in this report, the COG user must be aware of what is happening to particles in the random walk process. Over the years, many incorrect results were calculated and presented as valid information simply because Monte Carlo users did not take the time and effort to insure that all important parts of phase space were sampled. Frequently the user did not do his job simply because the Monte Carlo code being used produced a mountain of output that was just too much to comprehend. So, there is a problem of too much data or too little data. COG gives you three types of output to look at. The first is a simple balance table indicating the generation, movement, and reaction of particles. This comes without your asking for it and is described below in the the section **Explanation of Normal Code Output**.

The second type of output is some very simple pictures of parts of phase space indicating where events occur. The user must request this output, because each picture must be tailored to a specific problem. In the section **Additional Analysis the User May Request** (page 163, is an example of this type of problem analysis).

The third type of output contains information related to specific detectors and is explained in the chapter **Specific Results Using Detectors**, starting on page 167.

Explanation of Normal Code Output

Automatically included with every problem is a *summary of random walk events*. This is a summary of what happens to particles throughout the random walk phase of the problem. It is presented for each statistical region and represents information integrated over all times, energies, and angles. A representative sample of the computer output for an iron region in a water/iron transmission problem is shown below.

Region number 2

event	particle	# scoring	removal	addition	deposited E.
current-in	neutron	2914	0.	1.8679e+10	0.
	photon	573	0.	4.5968e+09	0.
current-out	neutron	2581	1.6545e+10	0.	0.
	photon	798	7.5000e+09	0.	0.
elastic	neutron	3203	2.0532e+10	2.0532e+10	2.8925e+07
(n,n'g)	neutron	19	1.2179e+08	1.2179e+08	5.7397e+06
	photon	19	0.	1.2420e+08	0.
(n,g)	neutron	333	2.1348e+09	0.	1.2414e+08
	photon	333	0.	3.7441e+09	0.
rayleigh	photon	35	2.9925e+08	2.9925e+08	0.
compton	photon	524	4.9439e+09	4.9439e+09	3.2889e+09
photoelectric	photon	407	1.6607e+09	3.9677e+08	1.8539e+08
pair production	photon	27	2.9874e+08	5.9749e+08	1.5628e+09
totals	neutron		3.9333e+10	3.9333e+10	3.5906e+07
	photon		1.4703e+10	1.4703e+10	5.0371e+09
	all				5.0730e+09

The information in this output should be self-explanatory. Events are indicated in the first column followed by the particular particle type considered. The number of scoring events follows—this value has no physical meaning to the problem solution, but it does indicate how many events in the Monte Carlo random walk occurred to contribute to this line of information. The number of scoring events does not include the particle statistical weight, so the value is not a real answer. However, if this number is small,

the results will have a large uncertainty. If it is large, the results will be more certain.

The last three columns contain results normalized to the source strength of the problem. The column labeled *removal* has the number of particles taken from the region. This may occur through leakage from the region or particles entering into collisions or random walk modifications. The column labeled *addition* includes leakage into the region, a fixed source, or particles coming out of a collision or random walk modification. The sum of the additions will equal the sum of the removals. In the last column, the amount of energy deposited within the region is given.

Each row lists a particular event or reaction that occurred within the region. There are about 100 different such events, so the example, or any real problem, will show only a few of them. The labels should properly identify each. Listing each reaction type greatly lengthens the summary tables, but we feel it is important that users not forget the physics associated with the various reaction types. Cross-section data are very important parts of any transport code, and it is easy to forget that they even exist.

To answer questions that may occur when looking at these summaries, we will just point out several facts:

- Any region that was specified in a problem and that had no events occurring in it will be identified by printing the region number and the titles, but it will have no additional data.

- Region number zero exists in all problems even if the user did not define it. It represents the default or undefined volumes in the problem. Generally, it is the void outside of defined regions.

- A *low-energy* event signifies that a particle has reached an energy less than the tabulated cross-section data.

- The number of scoring events goes with the particles entering a reaction or an event, not with the particles exiting from such events.

- A reaction labeled `C = number` refers to the ENDL library[1] number for a given type of reaction. All presently used `C` numbers within ENDL have the correct reaction definition built into COG. As new data are added to this library, however, COG

will have to be updated to include the correct definitions. Until that occurs, only the value of the C number will be indicated.

- If secondary particles are included in a problem, the sum of all deposited and leaked energy may be more than the total source energy.

Additional Analysis the User May Request

A simple summary of events does not always tell you everything you might want to know about the random walk. As an example, you may very well want to know where collisions are occurring within a specified time interval. Or, possibly, you are concerned with collisions of a given type of particle whose energy is within a fixed energy range. Now if the code divided all of phase space into small increments and printed results for each, there would be so much data the user would not look at any of them. Rather than do so, COG will allow you to draw a picture of collision sites within a specified range of phase space. This will let you know qualitatively what is happening in the random walk process.

The code will draw these pictures only when instructed to do so by the user. The instructions are contained within the ANALYSIS input block of data. Therefore, the existence of this block is a flag signaling that analysis pictures are to be drawn. Each picture request, and there may be any number, that the user desires, has the following information in it:

```
NEUTRON
PHOTON     xtl ytl ztl   xbl ybl zbl   xbr ybr zbr
.......
           thickness       {TITLE="......"}
           {ENERGY = elower  eupper}
           {AGE = agelower  ageupper}
           {WEIGHT = weightlower  weightupper}
```

The code will draw a picture indicating region boundaries, similar to the cross-sectional pictures of the geometry routines, with the top left corner at (x_{tl}, y_{tl}, z_{tl}), the bottom left corner at (x_{bl}, ybl, z_{bl}), and the bottom right corner at (x_{br}, y_{br}, z_{br}). The angle between the two lines defining the bottom and right sides of the picture should be 90 degrees. A volume is defined by two planes parallel to this plane—each one-half the thickness above and below this plane. All scatterings occurring within this volume will be counted. Limitations may be made to exclude scatterings with incident energies outside of a specified range, all

ages outside of a specified range, and all particles whose statistical weight is outside of a specified range, or any combination of these three items.

The following page illustrates such a picture drawn for a problem with a 0.2-MeV neutron source in water having an iron spherical shield around it. Note that the high-energy neutrons have many collisions close to the source and very few within the iron. Now part of this is understandable since the *slice* taken through the sphere is 2 cm thick. More scatterings occurring close to the source will be shown, since everything at a radius of less than 2 cm will be plotted while only about 15 percent of those occurring at 10 cm will be plotted.

The users will have to gain for themselves a *feeling* for the information contained on these pictures. A mental adjustment will have to be made for collision densities. The users will also mentally have to correct for the fact that the region boundaries indicated are on the middle plane of the volume being considered. Collisions above and below this plane are plotted as if projected on to the plane and may not be in the indicated region. The best way to consider this is to look at two such pictures, each drawn with a different modification to the random walk.

LOCATION OF COLLISIONS OF RANDOM-WALK PARTICLES

point 2 MEV neutron source in water with Fe shield
0.5 to 2.0 MEV neutrons
12/31/85 08:57:09 d

top left corner (-1.600e+01, 1.700e+01, 0.)
bottom left corner (-1.700e+01, -1.700e+01, 0.)
bottom right corner (1.700e+01, -1.700e+01, 0.)

thickness = 2.000e+00

particle type = neutron

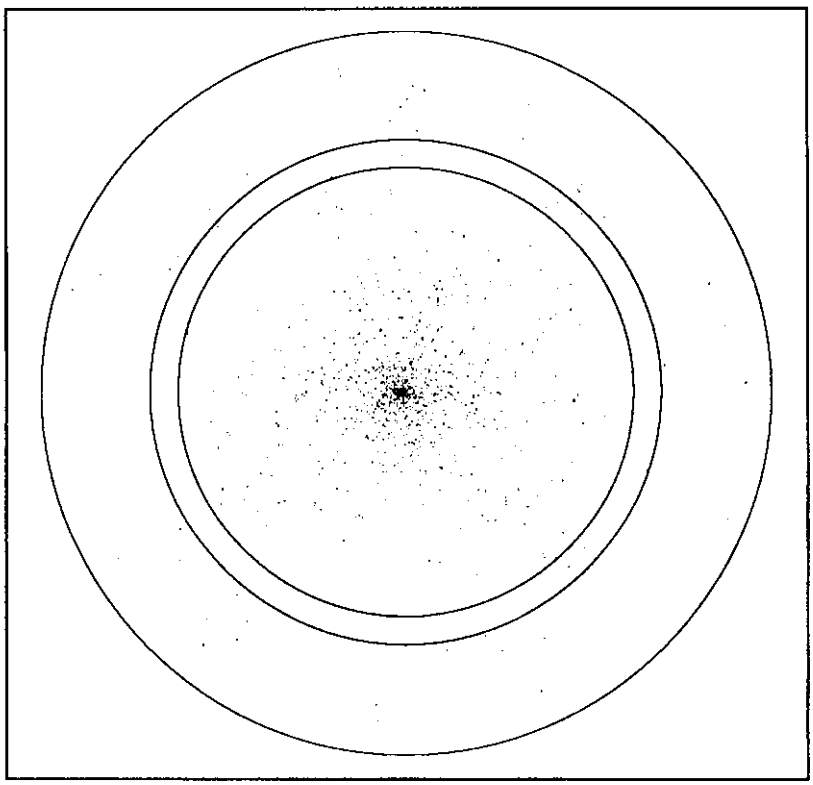
energy limits
5.000e-01 to 2.000e+00

time limits
-1.000e+99 to 1.000e+99

weight limits
0. to 1.000e+99

total monte carlo collisions within these limits
1425

number of these plotted at right
1425



The Random Walk

Specific Results Using Detectors

General Outline of Detector Specifications

An experimental physicist measures a radiation field by placing a physical *detector* at a given location and then reading the results of the detector. The detector may be a crystal that emits visible light when struck by a photon or a material that emits charged particles produced by neutron reactions. The detector may even be a piece of film that shows the tracks left by particles transversing it. We use similar terminology to describe a detector within the COG code. For the calculational physicist, however, the detector is a set of calculations performed on the results of the random walk to produce the same kind of answers that would be obtained experimentally.

If the results obtained from the random walk summary provide sufficient information to satisfy the requirements of a specific problem, detector calculations may be omitted from COG. Otherwise, the DETECTOR block of input information will be employed to specify any desired number of different detectors. Each defined detector is independent of the others, and each has the general input format:

```
NUMBER = id-number   {TITLE="....."}
         [type and associated data] {masks}
         {response-functions}  {differential-bins}
```

The definition of the second defined detector follows directly behind the first and so on.

The detector identification number is any BCD input with up to eight characters. It will be used to identify this specific detector and its results. There is no need to make the numbers, if they are numbers, consecutive or in any given order. The optional title associated with the detector is also for identification and is printed along with the calculated results.

Information on the type of detector desired must be provided. This informs the code as to the techniques to be employed when obtaining the detector answer and also provides any required physical data. In the section **Specific Types of Detectors** (page 168), the detector types currently available in COG are explained and the associated data needed for each are listed.

Specific Results Using Detectors

Each detector may have associated with it a set of *masks*. These have the effect of limiting the energy, age, or angles of particles to be scored to those within a given range. This provides a means of specifying that the detector has its own properties, such as collimation or minimum energy response, that are not built into the random walk problem.

The user may then specify a set of detector response functions dependent on particle energy, age, or angle. The effect of the response functions is to convert the calculated particle flux to a response—i.e., neutron and/or gamma flux to a biological dose, gamma flux to an electron production rate, etc. If not provided, the resulting answers will be in terms of particle-number flux.

Frequently, it is desired to know the response as a function of one or more of the basic scoring parameters—energy, age, or angle. Provision has been made to display these differential results as requested by the user.

To assist in picking a good set of parameters when modifying the random walk, various methods of obtaining the importance associated with the problem parameters have been included.

Each of these inputs is discussed in detail in the rest of the sections of this chapter.

The results calculated for each detector, with their associated graphical representations are, given in the output file of the code. An abbreviated output is also given in an ASCII file starting with the letters COGD. Comparison with the full output will allow the user to identify the meaning of each value. The user may disregard this file or may use it to convey numerical results to other programs.

Specific Types of Detectors

When a source particle is read from the source file, an event history record is started within the fast core memory for that particle. Each *event* in the random walk—like a boundary crossing, scattering, splitting, or Russian roulette game—is recorded in this record. This also includes all secondary particles produced by this original source particle. The printcut produced by the RETRACE option is an abridged version of this record (pages 195-197).

At the completion of each source particle, the code stops and analyzes the event history record once for each defined detector in the problem. The *score* resulting from this work is then kept for future output. The particular method of analysis is determined by the type of detector specified by the user. At the current time, only four different types have been built into COG. Additional types may easily be added to the code in the future, since the programming required only involves working with the event history record itself. Current programming does not save the event history after the analysis of a single source particle is completed.

Reaction Detectors

The simplest detector type, and many times the best, is a reaction detector. If the reaction detector is used, the code counts the number of collisions occurring within a specified volume. For each collision the code scores the incoming statistical weight divided by the total cross section and by the physical volume. Properly normalized, the total score is then the particle number flux averaged over the volume.

COG uses a slight variation of this method—rather than summing the real collisions, it sums the expected collisions along every path through the volume. If the defined volume is big in terms of mean-free paths, the two methods yield the same result. If the volume is *thin*, the method COG uses produces a better result, because there are more *scores* to sum.

The method of calculating the expected collisions is able to handle volumes with zero cross sections and still produce the correct fluxes. This method is best when you are satisfied with the average over a volume large enough to have a lot of particles going through it. If the volume becomes small and very few particles transverse it, the statistics become poor, and the answers are not too good. This latter problem can be solved by modifying the random walk to force particles toward the volume of interest. In any case, the results do not *blow up* because of mathematical assumptions, and the statistics properly reflect the real events being measured.

The required definition for this type of detector is:

```
REACTION    region    volume
```

Specific Results Using Detectors

where *region* represents the number of the statistical region defining the specific volume desired and *volume* is the proper volume, in the cube of the units of length being used in the problem, to go with this region. Note that the statistical region does not have to be all in one place in the problem geometry but may occur in several separated locations. The calculated result, in that case is the average over all these separate locations.

To insure that this type of detector produces the correct result, the user should insure that the defining region number occurs only in sectors that are bounded by other sectors containing a different region number. Note that the use of the connection OR in the geometry input produces a different sector as far as the code is concerned. If the defined region contains a void, this restriction can be ignored.

Boundary-Crossing Detectors

A boundary-crossing detector produces the average number flux on that area defined as being between two specified statistical regions. One way of looking at it is to take the reaction detector and shrink the volume so that it has zero thickness in one of the dimensions. The resulting score is then the incident particle weight divided by the cosine of the angle between the particle direction-of-motion and the normal to the surface being hit. The proper normalization, of course, has to be made. Unfortunately, the cosine goes to zero for particles that have a grazing collision with the surface. Dividing by zero causes bad things to happen within any computer code, and dividing by numbers approaching zero produce results that cause problems with statistics. To prevent this from happening, any time the cosine is less than 0.01 it is set to 0.005. There is some rationale for doing this, but it is only valid for certain geometries and not in general. We are saved only because, in most problems at least, almost no particles hit the surfaces with a grazing collision. If you have a case where you expect that this is a problem, we suggest you go back and try a reaction estimator.

The proper input for this detector type is:

BOUNDARY $region_1$ $region_2$ *area*

where the two region inputs represent the statistical region numbers of two adjoining regions in the geometry definitions and the *area* is the proper area between them.

If you use this detector type with a detector response function representing the cosine of the angle from the normal of the surface, you get results for particle current rather than flux.

Point Detectors

The reaction detector produces the average flux within some specified volume. There is no meaningful problem where it can be said that this value is equal to the flux at a known point. If we make the volume small enough so any point within it is a good representation of the desired point, the statistics become so poor that we do not know anything. With the boundary-crossing detector, there is one set of cases where the flux is known at a point. This, of course, is for one-dimensional spherical problems. For all other cases, we just can *not* assign the average over a surface to be the exact value at any specific point on the surface. These methods make easy and direct use of the random walk itself.

To obtain an answer at a real point in space, we will have to use a much more complicated procedure known in COG as a point detector—it has been called a next-flight estimator or a last-flight estimator at various times in the literature.[24] This method estimates the flux at a point by summing the contributions from each source point and from each scattering point. Each differential contribution is of the form:

$$d\Phi = S(\vec{r}\,') \; e^{-mfp} \; / \; (\vec{r}-\vec{r}\,')^2$$

Where:

$S(\vec{r}\,')$ The source at a point r' in space directed toward the detector position r. For a fixed source, this is normally easy to obtain. For a scattering source, this involves the probability of scattering through an angle defined by the initial direction of the particle and the direction toward the position of the detector. There is, of course, an energy associated with this scattering source.

mfp The number of mean-free paths that the scattered or source particle must pass through to go from the source point to the detector.

$\vec{r}-\vec{r}\,'$ The distance between the source and detector.

Specific Results Using Detectors

The flux at the point is the integral over all phase space of this expression. Of course, this is just the integral form of the Boltzmann equation. This is an eight-fold integral and is evaluated by the Monte Carlo method. The random walk produces scattering points that form the points at which samples are taken to evaluate the integral.

From the standpoint of running a problem, we now have to do the usual random walk and then add on top of it the additional calculations indicated above. Each of these involves going to the cross-section data and evaluating the probability of scattering in a given direction and, then, going to the geometry and finding the distance through each material when traveling between two points. This takes time—so much that the original random walk computing time can almost be forgotten.

One would be tempted not to use the method except for two facts. The first is that we really do get the answer at a point, and the second is that we get the maximum amount of data out of each collision and each source generation. For many shielding-type problems, the point detector is the most effective and efficient method of obtaining a result. It is also the method that can yield a result that is orders of magnitude away from a correct value and then label the result with a small statistical error.

To tell COG that you want a point detector, input the following:

```
POINT   x  y  z
```

where the point is located at (x, y, z).

There are two problems with the point detector—one is that parts of phase space are not sampled enough or at all; another is that collisions close to the detector itself produce large single contributions to the result. These, in turn, produce a very large statistical error.

The general thoughts have been that a point detector result should not be believed unless the statistical error was very large. In that case, you were to believe the result and disregard the error. We have had a difficult time trying to convince users of that fact. Now, after 15 years of looking at results, we are convinced that all the large statistical errors we have seen have been the result of under sampling rather than collisions close to the detector.

To provide additional information to the user about both these problems, COG prints in the output a list of each statistical region and the percentage of the total detector answer that came from scatterings in that region. In effect, this breaks the integration into spatial parts and tells you what is going on in each. One of these regional results will include the detector and will indicate the effect of close scatterings. Regions with few, or no, scatterings will warn you of under sampling problems. We reproduce below this table taken from a sample problem. The term FSD represents the fractional standard deviation associated with the response. This is one standard deviation divided by the response.

SEPARATED BY THE STATISTICAL REGION IN WHICH SCATTERINGS OCCURED

region	source particles	collisions	%response	fsd
4	5000	7998	56.300030	0.008
5	5000	338	18.204625	0.215
2	5000	822	16.445965	0.134
1	5000	27876	7.479862	0.026
6	5000	213	1.301805	0.206
3	5000	221	0.267713	0.307

As we mentioned before, another problem with the point detector is the large amount of time it uses. There are a number of ways this can be reduced—one of these has been built into COG. From what was said above, the code has accumulated results separated by statistical regions. In other words, it broke the integration over position into parts. Some of these parts contribute a large amount to the detector answer, some contribute almost nothing, and some have not been sampled enough to know what they contribute.

In the example just given above, the 7998 collisions in region 4, which included the fixed source, yielded 56 percent of the answer. This represents 21 percent of the collisions and, approximately, 21 percent of the work done to evaluate the point detector results. The error associated with this is very small, and one would almost always be satisfied with the calculated result. The scatterings in region 1, however, used 74 percent of the calculational effort and produced only 7 percent of the result. It would have been nice not to have sampled quite as often from this region. Regions 5 and 2 used only 3 percent of the effort and produced 35 percent of the result—they were under sampled. If we reran this problem, we

Specific Results Using Detectors

would like to take only a few samples from region 4 and from region 1 and concentrate our efforts on the remaining regions.

The following optional input to the definition of a point detector allows us to make adjustments to the amount of work that will be expended in each statistical region:

```
{LIMIT  [ REGION   ALL                    ]  n
        [ REGION   number                 ]
        [ REGION   number TO number       ]

        (REGION    ....    )  ....}
```

The last physically input value overrides any previous input value. This method has the effect of limiting the point flux estimations within a specified region to the first *n* source particles generated. The value of *n* is input to the code. The output of the code realizes that only a fraction of the work has been done and scales the results to that fraction. Needless to say, *n* must be equal to or less than the total source particles to be run in the problem. If no limit is input for a region, *n* is set to the total number of source particles. A value of *n* equal to zero implies that no contribution will be calculated from that region and the total result will not include the contribution from that region. *The use of any limiting will not allow the* COG *run to be terminated by the time or* END *conditions.*

We should again emphasize that the above technique presents results to you based on the total detector response. If you are interested in, say, time-dependent results, a particular time bin of interest to you may represent only a very small amount of the total response. In altering the problem with the limiting method just presented, you may be making the result in this time bin worse. To insure that you are doing the correct thing you may want to run the detector with a time mask that gives results only for this one time bin. The data presented to you will then correctly guide you toward a more optimum problem solution for that time.

Pulse Detector

A pulse detector simulates counting done frequently when performing real physical experiments. A crystal is used to measure gamma rays incident on it by observing pulses of light generated in the crystal by photon interactions. If a pulse is above a certain threshold energy, it is counted. If it is below that energy, nothing occurs. COG, when simulating a pulse detector, looks at the total gamma-ray energy (in MeV) deposited within a specified statistical region by all the reactions started by a single source particle. If this is greater than an input threshold energy level, a count of one is registered. Otherwise, no count occurs.
Note: Only a unity response function should be used with this detector type.

Experimentalists in the real world make this detection a little more complicated. There is a base detector like we have described above. There also are one or more additional detectors. We will count the base detector only if some logical set of events have occurred—for example, a count in the base detector and also in an additional detector (coincidence). We define the following three options:

a AND *b*	The counter *a* and another counter *b* must both have a count to generate a signal.
c OR *d*	There is a count either if a signal is detected in counter *c* or in counter *d*.
e NOT *f*	There is a count if there is a signal in *e* but no signal is present in *f*.

To activate this detector type, we must first provide the word PULSE and then define the base counter. It is assumed that the counter is defined by a single specific statistical region. In other words, statistical region 5 may define the base counter and nothing else. We then must provide a threshold energy (in MeV) to be used with this counter. Both are input to the code enclosed in square brackets [ ]. **Note:** *Only in this section*, the square brackets, [ ], *must* be input to the code.

A base detector defined by region 5 and with a 0.5 MeV threshold would be defined by:

```
PULSE    [5,0.5]
```

We will get one count for every source particle that deposits more than 0.5 MeV of gamma energy into this detector. If we add to the detection scheme another detector, defined by statistical region number 6 and having a 0.4 MeV threshold, and declare that there must be a pulse in both detectors to register the count in the base detector, the input would look like:

```
PULSE    [5,0.5]  AND  [6,0.4]
```

Sets of counters can be grouped together by enclosing them in square brackets. Thus we would count the pulse in base counter 5 only if we did not have a count in counters 32, 197, or 17, by specifying:

```
PULSE    [5,0.5] NOT [[32,0.4] OR [197,0.4] OR [17,0.4]]
```

The energy associated with the pulse detector is the energy of the gamma pulse in the base counter. Thus,the energy-dependent differential output is a pulse height presentation.

Do *not* use walk modifications with this type of detector if they would modify the particle weights after a collision in any of the defined counters.

Masks That Limit What a Detector Can See

Energy

The input to define an energy mask is:

{MASK-E *part.-type* e_{lower} e_{upper}

(*part.-type* e_{lower} e_{upper})

..... }

where *part.-type* is the usual input name for the desired particle—i.e., neutron, photon, etc.—and the energies (e) represent the lower and upper limits desired. The mask is used only for the detector with which it is defined and allows only

particles with energies between the given limits to be scored. The absence of an input for a given detector or for a given particle type always leaves the lower energy set at zero and the upper energy set to infinity.

Instead of adding data for all particle types to the same MASK-E statement, it is allowable to input a series of MASK-E statements, one for each specified particle type.

Time

Time masks are defined by:

$$\{\text{MASK-T} \quad t_{lower} \quad t_{upper}\}$$

These are used in the same manner as energy masks—no input implies no mask, and only particles with an age between the two input times will be scored.

Angle

If the desired mask is a function of only the angle between a reference direction and the incident particle direction, it is specified by:

$$\{\text{MASK-A} \quad \begin{bmatrix} u_{ref} & v_{ref} & w_{ref} \\ \text{NORMAL} & & \end{bmatrix} \quad \mu_{min} \quad \mu_{max}\}$$

where $(u_{ref}, v_{ref}, w_{ref})$ is the direction cosines of the reference direction and μ_{min} and μ_{max} are the cosines of the limiting angle. For the special case of a boundary-crossing detector, where the reference direction is desired to be the outgoing normal to the surface defining the boundary, the values of $(u_{ref}, v_{ref}, w_{ref})$ can be replaced by the word NORMAL.

If it is desired to have an angle mask defined by both a polar angle, as above, and also a azimuthal angle, the input would be:

$$\{\text{MASK-A*} \quad \begin{bmatrix} u_{ref} & v_{ref} & w_{ref} \\ \text{NORMAL} & & \end{bmatrix} \quad u_a \quad v_a \quad w_a$$

$$\mu_{min} \quad \mu_{max} \quad \theta_{min} \quad \theta_{max}\}$$

Specific Results Using Detectors

The azimuthal angle, θ, is input in degrees and (u_a, v_a, w_a) is the reference direction for this angle. **Note:** The two reference directions *must* be at right angles to each other.

It is not permitted to use both the MASK-A and the MASK-A* specifications with the same detector.

Half-Life

All discussions in this report up to this point assumed that when a neutron has a reaction and produces a secondary photon, the photon exits the reaction without a time delay. Actually, there is a small time delay; but it is so small that we neglect it. We have not considered the cases where the neutron excites the nucleus and the decay photons are not emitted until days, weeks, or even years after the initial event. These are problems of activation and are not really different from a standard problem except that the appropriate time decay must be considered. Cross sections for these events include activation data and the decay schemes known for most radioactive isotopes. These data are contained in the ACTL library.[25] A COG problem considering activation would use a special neutron cross-section library in which the secondary production data come from ACTL rather than ENDL.[1] A decay constant is associated with every secondary photon. The calculated results from a COG problem then must be sorted by half-lives and then multiplied by a time-dependent factor that includes the length of original irradiation and the time since irradiation.[26]

The half-life mask then sorts the calculated results based on the half-life of secondary photons. It has the form:

{MASK-HL *half-life*}

where *half-life* is the half-life desired in the units of the reference time. The code then accepts, for results of this detector, all decay constants calculated for half-lives from 0.975 to 1.025 of this input value.

At the time this report was written, an input cross-section library for use in COG has not been generated with the activation data. This work, like the albedo libraries, will be done when a user has a problem needing these features.

Detector Responses

Energy

Without additional input information to the COG reaction, boundary-crossing and point detectors will calculate the number of particles per square centimeter—regardless of the input measure of length. Sometimes this is what the user wants, but in most cases the quantity desired is the response to this number flux. Almost always, the conversion is energy dependent. The general form of specifying this is:

```
{DRF-E  part.-type  [response]
        (part.-type  [response]
        ..........................  }
```

where *part.-type* is the name of the associated particle, neutron, photon, etc.; the proper inputs for the response are given below. If desired, rather than stacking different particle type information one behind the other in one DRF-E statement, a series of DRF-E inputs, each containing only the information for one particle type, can be input to COG .

If no DRF-E inputs are given in the description of a given detector, all particles in the problem will have a response value of unity. If there is more than one type of particle specified in the problem and if one of these has a DRF-E input for it and the other types have not been specified, the unspecified type will be set equal to zero.

The response itself may be specified by any one of the following:

- The input word NUMBER-FLUX. This yields a response equal to one at all energies.
- The input word ENERGY-FLUX. This yields a response equal to the energy, in MeV, at all energies.
- The input word DOSE. Produces a result in REM. For neutrons with energies from 0 to 20 MeV, the built-in data are taken from the ANSI/ANS standard 6.1.1 (1977). From 20 to 200 MeV, the data are taken from Table 1.3-7 of the *Engineering Compendium of Radiation Shielding* (1968). For photons greater than 10 KeV, the conversion factors are from the ANSI/ANS standard 6.1.6

(1977). Photons less than 10 KeV will use a zero conversion factor and a code warning will be printed.

- The input ENERGY-DEPOSITION followed by the material number of one of the input materials in the MIX block will produce the energy deposition in that material with units of MeV per cubic centimeter.

- The input E-PRODUCTION followed by the material number of one of the input materials defined in the MIX block will produce an output indicating the electrons generated in that material by photon interactions. The energy and angle outputs for this detector will then represent the electrons being produced. Note that *no* electrons are followed when using this response and that additional photons and additional electrons generated by electron reactions will *not* be included. This option provides first estimates to those who are trying to calculate detector sensitivities.

- The input of EE-PRODUCTION followed by the material number produces the above-described output multiplied by the electron energy.

- A numerical description of the response function can be provided by the usual input of point-wise values, i.e.:

  e_1 $response_1$ e_2 $response_2$ .....

  where the response is zero above and below the input range.

Time

A response function as a function of time may be provided by the following input:

{DRF-T t_1 $response_1$ t_2 $response_2$ ....}

Where these are point-wise inputs. Like all other inputs of this type, the response will be zero above and below the input time-range.

Angle

Angle-dependent response functions are very similar to angle-dependent source specifications (pp 177). For a response dependent upon only one angle from a given reference direction, the input is:

$$\{\ \text{DRF-A} \quad \begin{bmatrix} u_{ref} & v_{ref} & w_{ref} \\ \text{NORMAL} & & \end{bmatrix} \quad \mu_1 \quad \text{DRF}_1 \quad \mu_2 \quad \text{DRF}_2 \ \ldots\ldots\ \}$$

Where the NORMAL input can be used only with a boundary-crossing detector. The remaining input represents point-wise data describing the response function and μ represents the cosine of the angle between the reference and the trajectory of the scoring particle.

For the case with a full double angle definition, the inputs take on the form:

$$\{\ \text{DRF-A*} \quad \begin{bmatrix} u_{ref} & v_{ref} & w_{ref} \\ \text{NORMAL} & & \end{bmatrix} \quad u_a \quad v_a \quad w_a$$

$$\mu^- \ \mu^+ \ \theta^- \ \theta^+ \ \text{DRF}$$

$$(\ \mu^- \ \mu^+ \ \theta^- \ \theta^+ \ \text{DRF}\ )$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \}$$

Where the given value of the response function is used within the entire range specified to go with it. Any part of the solid angle omitted from the input description automatically has the response function set to zero.

If desired, the full description can also be given by:

$$\{\text{DRF-A*} \quad \begin{bmatrix} u_{ref} & v_{ref} & w_{ref} \\ \text{NORMAL} & & \end{bmatrix} \quad u_a \quad v_a \quad w_a$$

$$\text{N} = \textit{number} \quad \text{DRF}_1 \quad \text{DRF}_2 \ \ldots\ldots\ \text{DRF}m\ \}$$

where *number* defines the COG standard equal solid-angle bin structure.

Obtaining Differential Results

Frequently users want to know something about their answer in terms of its energy, time, or angle dependence. (The variations in terms of the spatial coordinates is normally obtained by using detectors placed in different locations.) In COG , we refer to these as the differential outputs associated with each detector. We thus have requests for energy-dependent results, for time-dependent results, for angle-dependent results, or for time- and energy-dependent results, and so forth.

Each request is added to the inputs for each separate detector, so the inputs must be repeated if the same set of requests are desired for different detectors. As long as there is room in the machine, any number of requests may be made for each detector. This implies that more than one set of, say, time bins may be specified for the same detector.

The request for a single differential output is preceded by the word:

```
BIN
```

and then followed by the appropriate bins to be used. These may be time bins, in which case the input would look like:

$$\texttt{BIN} \quad \begin{bmatrix} \texttt{TIME} \\ \texttt{T} \end{bmatrix} = t_1 \quad t_2 \quad t_3 \quad \ldots$$

where the listed time values are the bin boundaries in either increasing or decreasing order. As is the case with all the bin structures, their range does not have to include the entire range of the specified variable.

Energy bins, which can include only one particle type, have the form:

$$\texttt{BIN} \quad \begin{bmatrix} \texttt{ENERGY} \\ \texttt{E} \end{bmatrix} = \begin{bmatrix} \texttt{NEUTRON} \\ \texttt{PHOTON} \\ \texttt{.....} \end{bmatrix} \quad e_1 \quad e_2 \quad e_3 \quad \ldots$$

Angle bins that are dependent on only the cosine of the angle between some reference and the incoming scored particle are specified by:

$$\text{BIN}\ \begin{bmatrix}\text{ANGLE}\\ \text{A}\end{bmatrix} = \begin{bmatrix}u_{ref} & v_{ref} & w_{ref}\\ \text{NORMAL}\end{bmatrix}\ \mu_1\ \ \mu_2\ \ \ldots$$

If desired, the reference direction may be specified by the word NORMAL, or by any method used in the path-stretching or angle-biasing techniques described in the chapter **The Random Walk** (see page 145).

Each of the above requests would produce a single output that is a function of just one output variable —that is, a one-dimensional (1-D) output.

If we wanted to see how the answer varied in both time and energy, we would specify:

$$\text{BIN}\quad \begin{aligned}\text{T} &= t_1\ \ t_2\ \ t_3\\ \text{E} &= \begin{bmatrix}\text{NEUTRON}\\ \text{PHOTON}\\ \ldots\ldots\end{bmatrix}\ e_1\ \ e_2\ \ e_3\ \ \ldots\end{aligned}$$

The output results would then have a set of energy bins for each time bin. It would have made no difference if the input had specified the energy bins first and the time bins last. There are a lot more individual bins for this two-dimensional (2-D) representation, and it may be necessary to run your problem longer to achieve meaningful statistics. Other cases of 2-D bins would be time plus single-angle bins and energy plus single-angle bins.

Another case of a possible 2-D output is where a full angle description is desired. This would be requested by:

$$\text{BIN}\quad \text{A*} = \begin{bmatrix}u_{ref} & v_{ref} & w_{ref}\\ \text{NORMAL}\end{bmatrix}\ u_a\ \ v_a\ \ w_a$$
$$\mu\ -\ \ \mu\ +\ \ \ \theta\ -\ \ \theta\ +$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$

Specific Results Using Detectors

This request is just like the source description (pages 171) or like the DRF-A* input (page 181).

It is also possible to specify this in terms of the standard bin structure:

$$\texttt{BIN A*} = \begin{bmatrix} u_{ref} & v_{ref} & w_{ref} \\ \text{NORMAL} & & \end{bmatrix} \; u_a \quad v_a \quad w_a$$

$$\texttt{N} = \textit{number}$$

Since both specifications inherently have two parameters, it is not necessary to give a second set of bin structures to have a 2D output.

Three dimensional outputs can be obtained by specifying any of the following:

```
BIN T= ...  E= ...  A= ...

BIN T = ... A*= ...
```

or,

```
BIN E = ... A*= ...
```

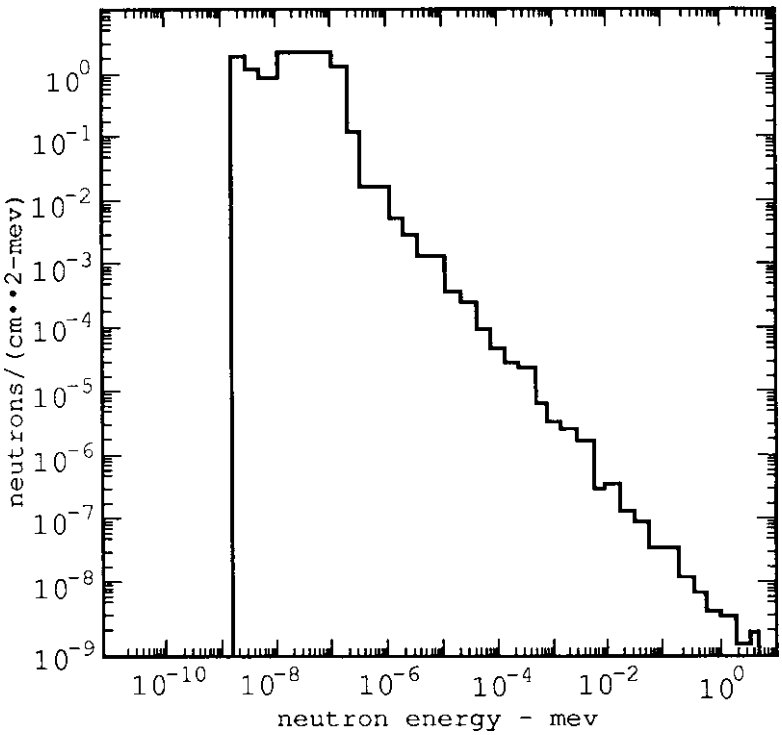
Four dimensional outputs are a combination of T, E, and A* specifications.

The differential outputs are always printed along with the complete definition of the bin structure. These outputs have units of response per second, or per MeV, or per steradian for the 1-D cases and the correspondingly complex units for the 2-D, 3-D, and 4-D cases.

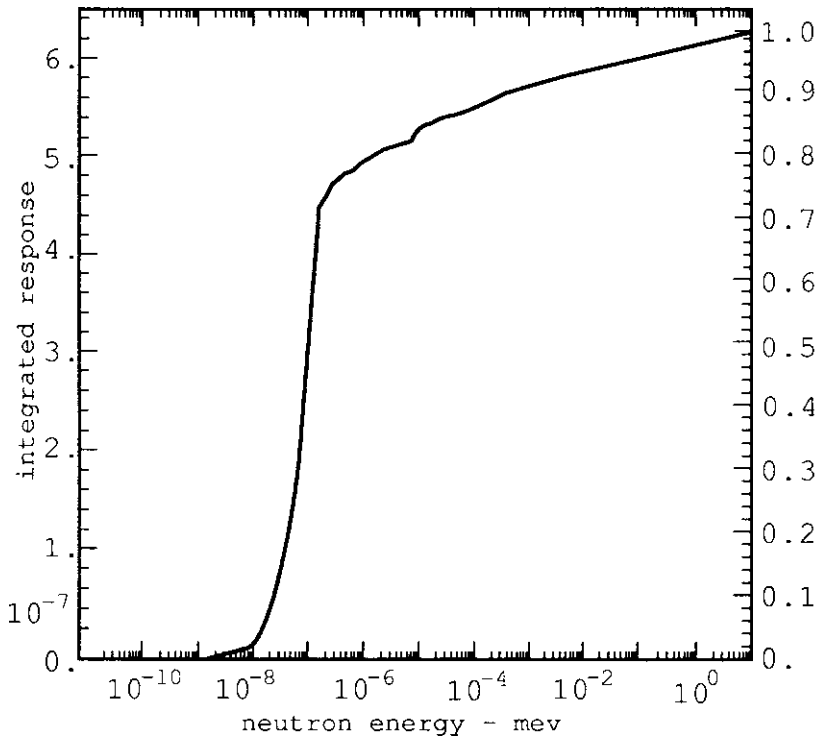
Where room allows on the printed line, we also include the standard deviation, the fractional standard deviation, the contribution made to the response from this single bin, and the integral response—i.e., total response from the lowest bin to the current bin. If the bin structure includes the full range of the differential variables, the final value of the integral response should equal the total response given for the detector.

Each requested 1-D differential output will also include three plots. The first will be the calculated quantity as a function of the differential quantity. The second will be the same, but it includes

dotted lines to indicate both plus and minus one standard deviation. The integral response is indicated on the third plot. The first and third plots are illustrated below.



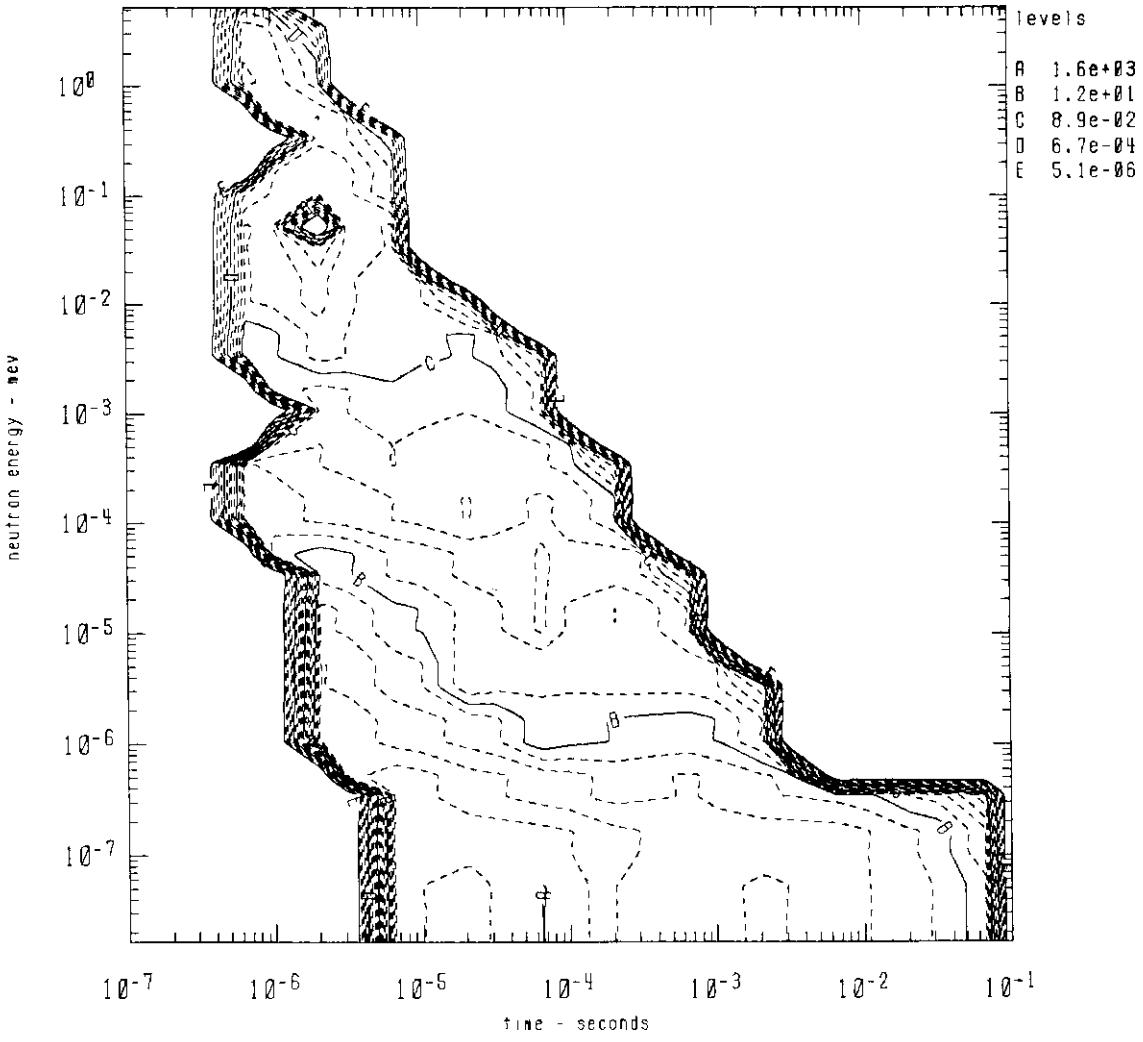
Specific Results Using Detectors



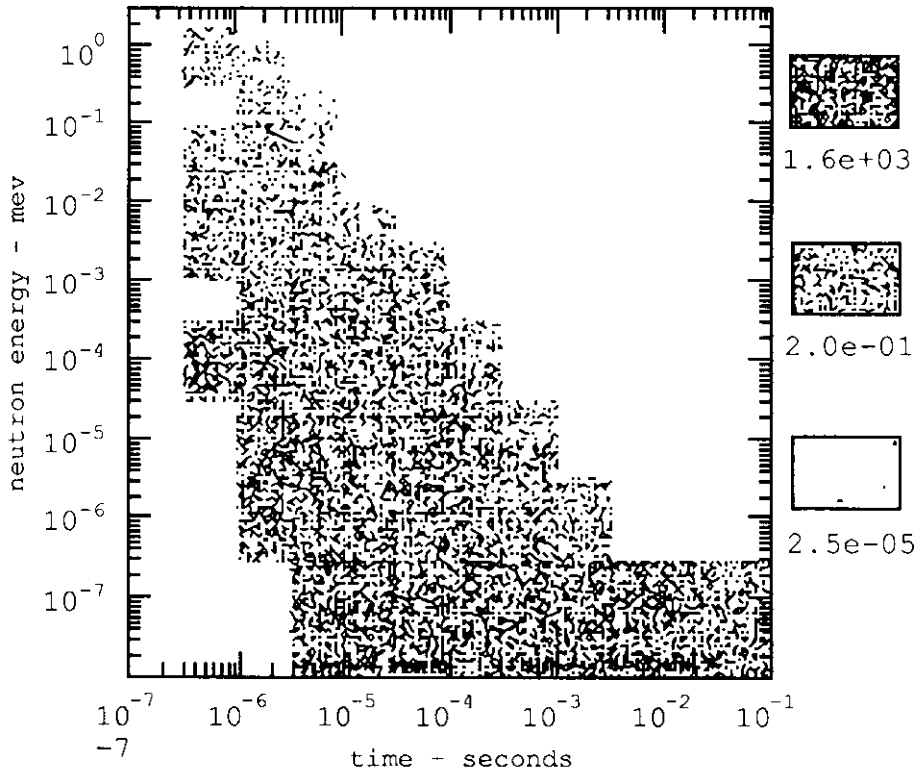
Two-dimensional outputs include three different plots to illustrate the calculated values. The first plot is a contour plot with the contours representing constant values of the differential response. Sometimes this produces useful information so you may gain a *feeling* for the results and sometimes it doesn't. In no case should the plotted values be taken to be exact, since the calculated values used to make these plots is a histogram and not point values.

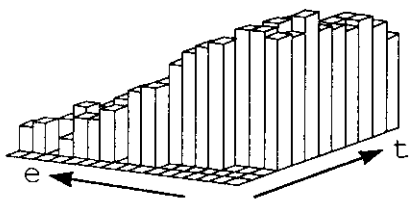
The second plot is a density representation of the calculated data. This may show the histogram nature of your differential bin structure. The third plot is a series of 3-D histograms. An example of each of the three types of 2-D plots are shown below.

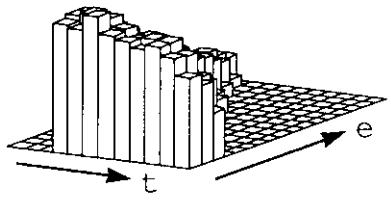
NOTE: contours are logarithmically spaced

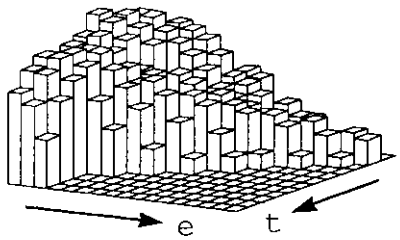


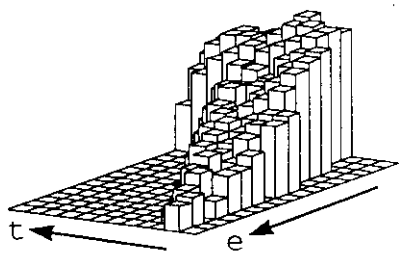
Specific Results Using Detectors











For 3-D and 4-D differential output, you can figure out how to draw your own plots. For this reason, and to accommodate those who wish to do other things with the output, all calculated values are written into an ASCII file and saved for future use. Viewing the ASCII file along with the regular COG output will readily identify the quantities contained therein.

You may wish to change the scaling or the labels on the plots that are made to reflect the specific problem being run more correctly. There are default labels setup within the code, and these may be appropriate. If they are not, the following will change them:

XCONV = *scale-factor*	Multiply the output values or the bin structure variable by the given factor before printing or plotting.
XMIN = *min-value*	Set the minimum scale value on the plots.
XMAX = *max-value*	Set the maximum scale value on the plots.
XLOG	Plot with a log scale.
XLIN	Plot with a linear scale—XLIN cannot be use with XLOG.
XLABEL = "..."	Enter the given label along this scale.

Note: The standard output for time has units of seconds and for energy the standard is MeV. These are not changed by inputs in the BASIC input block.

The above information is input after the last entry defining a bin structure if it applies to that structure or after all inputs in a given bin definition if it applies to the calculated quantity. The letter *X*

used above is replaced by the following letter to designate the quantity being defined:

T	Time
E	Energy
P	Cosine of polar angle
A	Azimuthal angle
Y	Calculated differential response

Following any inputs for the differential response, it is also possible to specify an additional title to be used with this bin structure by adding:

```
TITLE = "....."
```

The following would be an example of a single detector definition along with two differential structures.

```
DETECTOR
    NUMBER 1
    TITLE = "AVERAGE ENERGY FLUX IN LEAD RING"
    REACTION  28  197.321
    MASK-T  0 1.0E-6
    DRF-E  PHOTON  ENERGY-FLUX
    BIN
        T= 1-7 2-7 3-7 4-7 5-7 6-7 7-7 8-7 9-7 1-6
           TCONV=1.0E+6 TMIN=1-7 TMAX=1-6 TLOG
           TLABEL="TIME - MICROSECONDS"
        YLIN YLABEL="PHOTONS/CM**2-SEC"
    BIN
        E=PHOTON .05 .1 .5 1. 2. 3. 4. 5. 6.
        T= 1-7 2-7 3-7 4-7 5-7 6-7 7-7 8-7 9-7 1-6
```

Calculated Importance Results

In the chapter **The Random Walk** we talked about the meaning of importance when we try to bias a Monte Carlo problem. Throughout most of the inputs to COG, there have been optional importance values specified. The big problem, of course, is to know what values to input to the code.

For a particular detector, we have already seen two different importance outputs that are determined by the problem run itself. The first of these comes about by keeping track of the total response by the initial source increment. This provides an easy way to go

back and make more particles in the increment that provides the most answer. The second keeps track of the total response by the path number in the WALK-SOURCE option. With this information, more particles can be forced to follow the path that leads to the largest contribution to the total response.

The concept of differential bins can give rise to importance information if we change the meaning of what is attached to the bin structure. For an answer, we scored the result at the detector in terms of the energy of the particle reaching the detector. This gave us a differential result that showed what energy the particles had when they scored. If we change the meaning of the energy term to, say, the energy that the initial source particle had and then score it in a set of differential bins, we now will have the importance as a function of source energy.

In the same way we could use the initial age of the source particle or its initial direction. We can, therefore, obtain the necessary importances that go with the source provided, of course, that we can get a source particle to score at the detector.

In the same way, we can obtain the importances of particles exiting from a scattering collision in a given region. This is done by scoring in the differential bin structure according to the energy, age, and direction of particles leaving collisions in the region rather than by particles reaching the detector.

The calculation of importance is then identical to what we have just described for the bin inputs except for:

- The word `BIN` or `BINS` is replaced with the word `IMPORTANCE` or `IMP`.

- If we want to calculate the importance of the source, we then go ahead with the regular inputs for bins.

- If we want to calculate the importance of scattering in a given region, we add the word `REGION` followed by the region number and then go ahead with the regular inputs for bins.

The outputs will now be labeled as importances and may aid you in the correct biasing of subsequent problem runs.

Specific Results Using Detectors

Special and Rarely Used Items

Albedo Specifications

An albedo surface may be defined in the geometry specifications by using the special boundary definitions illustrated on page 81. Part of this input is a number for identifying a specific albedo. To connect these numbers to numerical data defining the albedo, the ALBEDOIN block of inputs must be present. The data needed as input to this block are the set of the albedo numbers as used in the problem and the name of the file on which the data defining each are to be found. Note that only one albedo definition is given in each file. The inputs take the form:

albedo-number$_1$ *file-name*$_1$ {*albedo-number*$_2$ *file-name*$_2$} ...

A maximum of 10 albedos may be input in a single problem.

Each albedo file is a binary file and contains the following:

- Lead data for file.

Position	Data
0 – 19	BCD description of the file
20	Number of words in data describing albedo
LSTART(9)	Starting position for particle type *itype* (1 to 9) (*itype* = 1 for neutrons and = 7 for photons)

Special and Rarely Used Items

- Lead data for each incident particle type.

Symbol	Length	Data
NIE	1	Number of incident energy groups.
EI	NIE+1	Boundaries (MEV) for each of the groups, in increasing order.
NIA	1	Number of incident angle groups.
CI	NIA+1	Boundaries (cosine of the angle) for each of these groups in increasing order. CI(1)=0, CI(NIA+1)=1.0
PRET	NIA*NIE	Probability of return for an incident particle in a specified energy and angle bin.
LINC	NIA*NIE	Pointer to return data for an incident particle in a specified energy and angle bin.

- Data describing the returned particle.

Symbol	Length	Data
NRE	1	Number of return energy groups.
ERU	NRE	Upper boundaries (MeV) for each energy group.
ERL	NRE	Lower boundaries (MEV) for each energy group.
IRTY	NRE	Particle type for each energy group.
NRA	1	Number of return angle groups.
CR	NRA+1	Return angle boundaries.
APD	NRA*NRE	Accumulative probability for hitting in specific angle and energy bin.

Needless to say, we have given you the format for these files because we do not have any albedo files to give to you. We have neither had an occasion to run an albedo problem and, therefore, have not generated these files nor have we debugged COG for albedo operation. When someone needs these options, we will be glad to work with them through this part of the code. We would expect that the basic physical data needed to define an albedo would be obtained by running a COG problem and post-processing the output.[27]

The Retrace Option

Occasionally you would like to know just what a specific random walk looks like. This means that you will have a printed copy of the original source specification in position, angle, energy, and time and then have similar printed data for all boundary crossings, scattering collisions, and walk modifications. Because the detector results indicate the 10 highest scoring particles in a problem run, you may want to look at some of these to know why they are large. The RETRACE option allows you to secure this information.

To use the RETRACE option, you must first run a COG problem and save the source files. These files, if you remember, contain the starting random numbers at the beginning of each source particle. You must then rerun the same problem specifying in the source block that you will use this existing source file. This block will then look similar to:

```
SOURCE    NPARTICLE=7000   FILE=COGSQTV0
```

No other information is necessary for the source description. If you wish to leave the original description in the problem, the code knows enough to skip over it.

The RETRACE block contains a list of the source particles (by number) that you wish to have printed out. These numbers are just the numbers assigned to the original source particle. The first is number one, the second is two, etc. A typical block input for this option then looks like:

```
RETRACE    7    298    4071    18    5099
```

Special and Rarely Used Items

With this option, the COG problem will only run the specified particles. The output for a single particle is shown below.

```
1  particle taken from initial source specifications
      x=-2.395419e+01  y= 5.729205e-02  z= 2.197097e-01
      u= 9.874965e-01  v= 1.376215e-01  w=-7.688353e-02
      age= 0.
      photon       E= 1.170000e+00  vel= 1.180285e+10
      statistical weight= 1.000000e+00
      number of collisions=     0
      material number=    4  region number=    4
      density factor= 1.000000e+00  total cross section= 1.221330e+00
      mean-free-paths left to travel= 2.001582e+00
      distance to next boundary 3.824407e-01

2  crossed a boundary between two different regions
      x=-2.357653e+01  y= 1.099241e-01  z= 1.903063e-01
      age= 3.240239e-11
      material number=    3  region number=    3
      density factor= 1.000000e+00  total cross section= 3.867529e-01
      mean-free-paths left to travel= 1.534495e+00
      distance to next boundary 6.305303e-02

3  crossed a boundary between two different regions
      x=-2.351427e+01  y= 1.186016e-01  z= 1.854586e-01
      age= 3.774458e-11
      material number=    6  region number=    6
      density factor= 1.000000e+00  total cross section= 1.922228e-04
      mean-free-paths left to travel= 1.509983e+00
      distance to next boundary 2.305251e+01

4  crossed a boundary between two different regions
      x=-7.500000e-01  y= 3.291123e+00  z=-1.586900e+00
      age= 1.990875e-09
      material number=    5  region number=    5
      density factor= 1.000000e+00  total cross section= 1.772116e+00
      mean-free-paths left to travel= 1.505552e+00
      distance to next boundary 1.518993e+00

5  enters into a collision
      x= 8.895621e-02  y= 3.408043e+00  z=-1.652218e+00
      age= 2.062856e-09

6  exits from a collision
      u= 9.880605e-01  v= 2.268149e-02  w=-1.523874e-01
      photon       E= 1.145205e+00  vel= 1.180285e+10
      number of collisions=     1
      density factor= 1.000000e+00  total cross section= 1.804352e+00
      reaction number=   72     compton
      mean-free-paths left to travel= 1.645095e-01
      energy deposition times statistical weight= 2.479500e-02
      distance to next boundary 6.690317e-01
```

```
 7  enters into a collision
       x= 1.790414e-01  y= 3.410111e+00  z=-1.666112e+00
       age= 2.070580e-09

 8  exits from a collision
       u= 9.930984e-01  v= 1.152486e-01  w= 2.175772e-02
       photon       E= 1.097347e+00  vel= 1.180285e+10
       number of collisions=     2
       density factor= 1.000000e+00  total cross section= 1.871266e+00
       reaction number=   72     compton
       mean-free-paths left to travel= 9.621324e-01
       energy deposition times statistical weight= 4.785811e-02
       distance to next boundary 5.749266e-01

 9  enters into a collision
       x= 6.896540e-01  y= 3.469368e+00  z=-1.654925e+00
       age= 2.114143e-09

10  exits from a collision
       u=-3.879478e-01  v= 5.292757e-01  w= 7.545619e-01
       photon       E= 8.494000e-02  vel= 1.180285e+10
       statistical weight= 9.680000e-01
       number of collisions=     3
       density factor= 1.000000e+00  total cross section= 5.803799e+01
       reaction number=   73     PE
       mean-free-paths left to travel= 4.023242e-01
       energy deposition times statistical weight= 1.015036e+00

11  entered into a russian roulette game

12  killed by losing a russian roulette game
       statistical weight= 0.


13  particle taken from secondary bank
       x= 6.896540e-01  y= 3.469368e+00  z=-1.654925e+00
       u= 9.258316e-01  v=-6.054500e-02  w= 3.730550e-01
       age= 2.114143e-09
       photon       E= 3.066000e-03  vel= 1.180285e+10
       statistical weight= 2.904000e-02
       number of collisions=     3
       material number=    5  region number=    5
       density factor= 1.000000e+00  total cross section= 6.705906e+04
       reaction number=   73     PE
       mean-free-paths left to travel= 6.049676e-01

14  entered into a russian roulette game

15  killed by losing a russian roulette game
      statistical weight= 0.
```

Color Specification for Special Options

The automatically set colors, when using the color picture option in the geometry specifications, may not be exactly what you desire. These are preset default values that work reasonably well, but no true artist would work without mixing his own colors and casting shadows appropriate to the picture being drawn. The COLORSET block of inputs allows the user to make such choices.

There are six different inputs in this block that may be used in any order. The six inputs relate to light position, color of a specified color number, surface color characteristics, color assignments, shadow casting, and picture resolution.

Light Position

The following statement defines the position of the light source within a given problem:

$$\left\{ \texttt{LIGHT} \quad d \ \theta \ \phi \quad \left[\begin{array}{l} \texttt{FIXED or FIX or F} \\ \texttt{RELATIVE or REL or R} \end{array} \right] \right\}$$

The position is measured in polar coordinates, a distance d away from the center of the picture and with an azimuthal angle θ and a polar angle ϕ. For the input FIXED, the light source stays in one fixed position within the geometry even as the viewer changes position. Theta (θ) and phi (ϕ) are measured in the picture's coordinate system the same as the viewer's position is specified in the original picture request. For the input RELATIVE, the light is fixed to the viewer's position—similar to a flash gun attached to a camera. Theta and phi are measured from the direction determined by the viewer's position.

The built-in default is:

```
LIGHT   10000   20   10  RELATIVE
```

Color Associated with a Specified Color Number

The inputs assign a physical color to go with a given color-number. The form of each input, and there may be more than one input of this kind, is:

{ PALETTE *color-number red green blue* }

The *color-number* is an integer in the range of 0 to 50 and will be used in the PAINT inputs to attach this color to a given material, region, or sector number (see page 201). The *red*, *green*, and *blue* inputs represent the amount of each of these colors present in the mixed physical color. Each may take any value from 0 to 1.

The built-in default colors are:

Color, number	Red	Green	Blue	Color, number	Red	Green	Blue
0	0	0	0	18	0	1	1/2
1	1	1	1	19	1/2	0	1
2	1	0	0	20	1	0	1/2
3	0	1	0	21	1/2	1	0
4	0	0	1	22	0	1/2	1
5	1	1	0	23	1	1/2	1/2
6	1	0	1	24	1/2	1	1/2
7	0	1	1	25	1/2	1/2	1
8	0	1/4	0	26	1	3/4	0
9	0	1	1/4	27	0	1	3/4
10	1/4	0	1	28	3/4	0	1
11	1	0	1/4	29	1	0	3/4
12	1/4	1	0	30	3/4	1	0
13	0	1/4	1	31	0	3/4	1
14	1	1/4	1/4	32	1	3/4	3/4
15	1/4	1	1/4	33	3/4	1	3/4
16	1/4	1/4	1	34	3/4	3/4	3/4
17	1	1/2	0	35-50	0	0	0

Color number 0 results in black, number 1 in white, number 2 in red, etc.

Special and Rarely Used Items

Surface Characteristics of Colors

How a surface appears depends not only on the color representing the material of the surface but also on the amount of ambient light, the diffuse reflection, the specular reflection, and how sharp the highlights are. These parameters determine how dark a surface is when it is completely in shadow, and they determine the appearance of a well lighted surface—whether it is yellow as a bright new shiny car or yellow as a duck's feathers. The following input sets all the surface characteristics in the problem to the same values:

$$\left\{ \begin{bmatrix} \texttt{COEFFICIENT} \\ \texttt{COEFF} \end{bmatrix} \; a \quad d \quad s \quad h \right\}$$

where:

- a is the ambient coefficient.
- d is the diffuse coefficient.
- s is the specular coefficient.
- h is the highlight coefficient. h is greater than or equal to zero with larger values yielding very *sharp* highlights.

By definition, $a + d + s = 1.0$

The built-in coefficients are:

```
COEFFICIENT    0.4    0.6    0.0    0.0
```

Full Color Assignment

The following input assigns color-numbers to material numbers, region numbers, or sector numbers and then goes on, if desired, to define the characteristics of each of these completely.

{ PAINT *m color-no.* $((a,d,s,h)_{red}\ (a,d,s,h)_{green}\ (a,d,s,h)_{blue})$ }

Where:

m is the material/region/sector number as used in the COG problem.

color-no. is the number, as described above, that will be used with the material/region/sector number.

(a,d,s,h) are the parameters defined in section **Surface Characteristics of Colors**, above, but are given for each of the three primary colors.

The default input of the code assigns the color number to the same valued material/region/sector number and does not override the values of *a, d, s*, and *h* as set by the default COEFFICIENT input.

Shadow Casting

Casting, or not-casting, shadows in the calculated picture is determined by:

{ [SHADOW / SHADOWS] ON or OFF }

The default is OFF.

Special and Rarely Used Items

Picture Resolution

When drawing a color picture, the picture area is divided into n x n pixels; and each of these is evaluated as to material, color, and intensity. If the code realizes that it is missing some of the detail in the picture, it will double the resolution to $2n$ x $2n$ pixels. This process continues, if necessary, until some upper limit is reached. Of course, each doubling results in approximately four times the original running time but does give a better quality picture. To control this process, it is possible to input the limits allowed:

$$\left\{ \begin{bmatrix} \text{RESOLUTION} \\ \text{RES} \end{bmatrix} \; n_1 \;\; n_2 \right\}$$

where n_1 is the initial number of intervals and must equal 120, 240, 480, or 960. n_2 is the final number of intervals and must be equal to or greater than n_1 and must take values of, again, 120, 240, 480, or 960.

The default values are $n_1 = 120$, $n_2 = 960$.

Criticality Option

The option for criticality is turned on by the inclusion of the CRITICALITY block of inputs as specified below. You should note that the fixed-source input (SOURCE block) is then not needed and, if provided, will be skipped over. Other options that need the input source file will not be operational for this type of problem—these include the RETRACE option and the correlated sampling technique.

It is believed that all other input options will function but, of course, many are not applicable for criticality work. Results for fluxes, currents, and detector responses may be obtained by specifying the usual detector inputs. These are obtained after the first (NFIRST-1) batches are thrown away. The use of the CRITICALITY block automatically sets the code to operate with neutrons (in case you forgot) and also specifies the inclusion of delayed neutrons. If you desire to calculate gamma fluxes from the gamma rays produced by fission, include PHOTON as a particle type in the BASIC block of inputs.

The inputs to the CRITICALITY block include the following quantities. The symbol # indicates input numbers.

NPART = #	Number of particles per batch in the initial batch; set to 500 if omitted. Subsequent batches will have approximately this number of particles.
NBATCH = #	Maximum number of batches that will be run; it is set to 200 if omitted.
SDT = #	Termination of a problem will occur after the first 20 batches when 5 consecutive batches have an associated standard deviation less than this value. It is set to 0.005 if not input.
NFIRST = #	First batch that will be used to calculate the average multiplication, fluxes, and leakage. It is given a value of 6 if omitted.
NORM = #	Normalization of output flux and related quantities—in fissions/second. The default is 1.
NSOURCE = #	Number of initial point sources used to define the starting source in the first batch. Identifies the number of (x,y,z) values provided below. Must be input.
(x, y, z)	NSOURCE sets giving the position of the starting sources. The NSOURCE sets must follow, in the inputs, directly behind the specification of NSOURCE.

Special and Rarely Used Items

Putting it All Together— An Example Problem

Description of the Example Problem

The last example problem we put in a manual was so complicated that none of the readers was ever able to understand it. The manual before that had a simpler problem but one that produced an obviously wrong result. Only two readers, out of hundreds, in the last 15 years spotted that the results were incorrect.

This time we are going to present a very simple problem that yields a correct result. But after all, the only reason to illustrate a problem is to show you what a running problem input looks like. We make no attempt to show you all aspects of all possible problem inputs—these will come in later volumes.

Assume we are back in the good old days of being a college student. It was a beautiful warm spring day and rather than go to our nuclear engineering laboratory, we went off on a picnic. Now, we are faced with writing up an experiment that we didn't do. Instead of doing the work in the laboratory, we will do it by calculating it at our computer.

What we were to do was fairly simple—there is a cobalt-60 gamma-ray source and a Geiger-Müller tube, and we are to count gamma rays. To made it more interesting, there are lead bricks to put between the two.[28]

The experiment was to be performed in a room 30 feet long by 20 feet wide and with a 10-foot ceiling. All the walls and the ceiling and the floor are formed of 12-inch thick concrete and, for purposes of calculation, we may forget doors, heating ducts and other minor features.

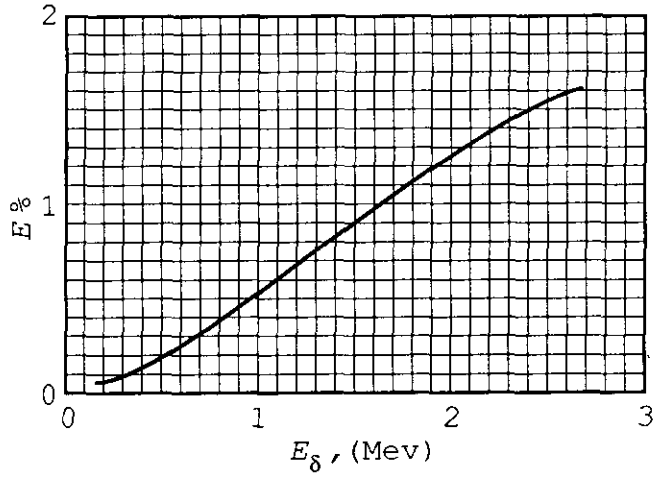
The concrete has a specification represented by the material CNCRT2 in the COG library. There is a single table at the center of the room with dimensions of 3 feet wide by 5 feet long and oriented with the longest length coincident with the longer dimension of the room. The table is made of wood (white oak, 12% moisture) 3 inches thick and with its top 36 inches from the floor. Four wood legs, each 4 inches square, are located at the corners.

An Example Problem

The gamma-ray source is enclosed in a cylindrical aluminum holder 1 inch in outside diameter and 1 inch in outside length and with 1/16-inch thick walls. This is oriented with the axis perpendicular to the table top and is held by a 0.25-inch diameter aluminum rod so the center of the cylinder is 12 inches above the table and 24 inches to the left of the center of the table.

The cylindrical holder contains cobalt metal that has been activated so as to have a source strength of 1 mcurie (3.7×10^7 disintegrations per second). Each disintegration yields one 1.17-MeV and one 1.33-MeV gamma ray.

A Geiger-Müller (G-M) counter is located 24 inches to the right of the center of the table—i.e., there are 4 feet between the counter and the source. The counter has an aluminum cathode and a response efficiency given by the curve below. The count rate of the detector is given by the product of the gamma flux at the detector, the response efficiency, and the detector area. We will assume that the area is 1 square centimeter. Note that we are roughly talking about 1 count per second for a flux of 500 gamma rays/square centimeter-second. Dead time corrections need not be considered; and the detector need not be modeled, because its response includes the actual detector geometry.



Lead bricks are piled in the center of the table to give a stack 16 inches high by 8 inches wide and 1.5 inches thick (the thickness is measured along the direction measured between the source and detector).

Forget all other items in the room (including yourself).

What count rate does the G-M counter register?

Now if you really want to see if you understand the COG input, go ahead and set the problem up and try running it without looking at the rest of this chapter. To save you a fairly large amount of time, we will give you the most difficult piece of needed information—the composition of the wood in the table:

carbon	0.32 grams/cubic centimeter
hydrogen	0.04
oxygen	0.30
nitrogen	0.07
water	0.04

If you succeed in running the problem without error comments, and you get an answer of about 0.26 counts/second, give yourself a gold medal and skip to the section **Discussion of Results** at the end of this chapter (page 243).

Code Inputs

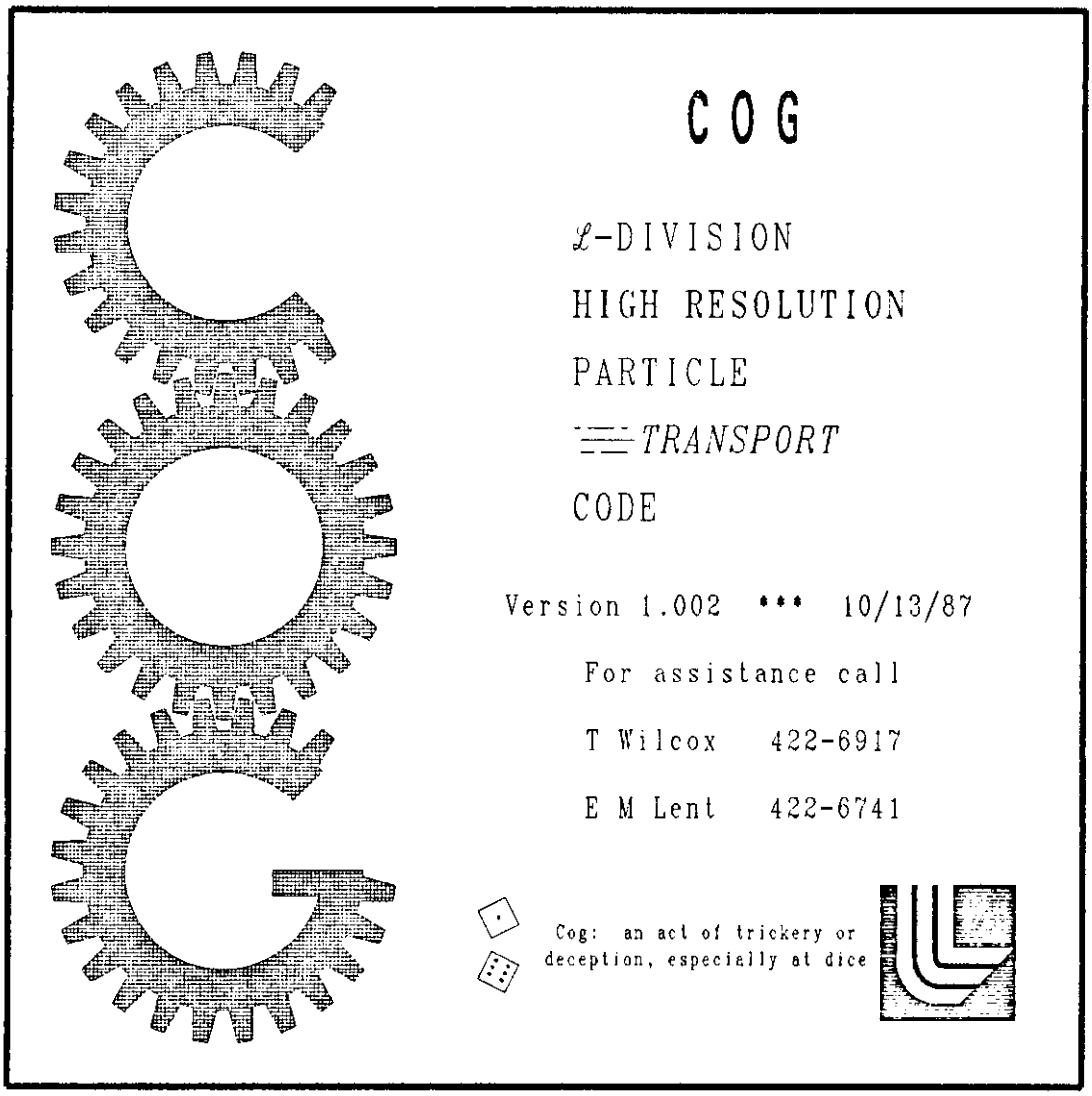
The first part of the COG output reproduces the problem's input. We have added comment cards to the input so that you may understand what the surfaces represent and how the problem is put together. There doesn't seem to be any need to explain these items a second time. It would be wise to go through these inputs sketching out the geometry and understanding the input values.

This is an analog problem with one exception - photons are killed by using the WALK-COLLISION method when they reach an energy of 100 KeV. Since the detector's response is zero below this energy. there is no reason to follow photons below this point.

Code Outputs

The entire problem output is reproduced on the following pages.

An Example Problem



```
COG  --  LISTINGS OF INPUT CARDS          10/14/87     15:36:18  h          examp

EXAMPLE PROBLEM --- Co-60 Source with 1.5 inches of Lead


BASIC  photon  inch


MIX
     mat=1 concrt2 2.51
     mat=2 c .32  h .04  o .30  n .07  h2o .04 $ white oak, 12% moisture
     mat=3 al 2.7
     mat=4 co 8.85
     mat=5 pb 11.4
     mat=6 air .00129


GEOMETRY
     sector 1 walls -1 +2
     sector 2 table -3 or -4 or -5 or -6 or -7
```

An Example Problem

```
     sector 3 holder -8 or -9 +10

     sector 4 cobalt -10

     sector 5 lead   -11

     fill   6

     boundary vacuum +1


     picture cs m -192.5 0 81.5  -192.5 0 -59.5  192.5 0 -59.5

     picture cs m -36 0 12  -36 0 -49  36 0 -49


SURFACES

      1 box 384 264 144 tr 0 0 12 $ outside room walls

      2 box 360 240 120 tr 0 0 12 $ inside room walls

      3 box  60  36   3 tr 0 0 -13.5 $ table top

      4 box 4 4 33 tr -28 -16 -31.5 $ table leg

      5 box 4 4 33 tr -28  16 -31.5 $ table leg

      6 box 4 4 33 tr  28  16 -31.5 $ table leg

      7 box 4 4 33 tr  28 -16 -31.5 $ table leg

      8 cyl .125  .5 12 tr -24 0 0 -24 0 -120 $ supporting rod
```

```
     9 cyl .5   -.5 .5 tr -24 0 0 -24 0 -120 $ source container
    10 cyl .4375 -.4375 .4375 tr -24 0 0 -24 0 -120 $ source
    11 box 1.5 8 16 tr 0 0 -4 $ lead blocks


DETECTOR
     number 1
     point 24 0 0
     drf-e photon  0.1 0.      0.2 8.0e-4  0.3 1.0e-3  0.4 1.6e-3
                   0.5 2.1e-3  0.6 2.6e-3  0.7 3.3e-3  0.8 4.0e-3
                   0.9 4.5e-3  1.0 5.1e-3  1.1 5.9e-3  1.2 6.6e-3
                   1.3 7.2e-3  1.4 8.0e-3
                $ GM tube sensitivity for Al cathode - p183. Bieuler & Goldsmith


SOURCE
    nparticle=5000
    define e=1  photon line 1.17 1.0  1.33 1.0
    define p=1  cylinder -24 0 -.4375  -24 0 .4375  .4375
    define t=1  steady
```

An Example Problem

```
    increment 7.4e+7  e=1  p=1  t=1


WALK-COLLISION

    photon region all  energy 0. to 0.1  0.0


END
```

BASIC PARAMETERS

units of length - inch

units of time - second

units of energy - mev

maximum running time - 1.0000e+08 minutes

random number seed - 0000003126215420447703B

particle types to be followed: photon

MIXTURE DEFINITIONS

material number	physical temperature	component name		specific gravity
1	20.0 C	concrt2		2.510e+00
			total	2.510e+00
2	20.0 C	c		3.200e-01
		h		4.000e-02
		o		3.000e-01
		n		7.000e-02
		h2o		4.000e-02
			total	7.700e-01
3	20.0 C	al		2.700e+00

An Example Problem

				---------
			total	2.700e+00
4	20.0 C	co		8.850e+00
				---------
			total	8.850e+00
5	20.0 C	pb		1.140e+01
				---------
			total	1.140e+01
6	20.0 C	air		1.290e-03
				---------
			total	1.290e-03

```
GEOMETRICAL SURFACE SPECIFICATIONS          unit length is one inch

  Surface Number     1      box
    center of box is at x',y',z' coordinate center
    length in x', y', and z' directions       3.84000e+02  2.64000e+02  1.44000e+02
    T/R data
      center of (x',y',z') system   ( 0.          .  0.          .  1.20000e+01)

  Surface Number     2      box
    center of box is at x',y',z' coordinate center
    length in x', y', and z' directions       3.60000e+02  2.40000e+02  1.20000e+02
    T/R data
      center of (x',y',z') system   ( 0.          .  0.          .  1.20000e+01)

  Surface Number     3      box
    center of box is at x',y',z' coordinate center
    length in x', y', and z' directions       6.00000e+01  3.50000e+01  3.00000e+00
    T/R data
      center of (x',y',z') system   ( 0.          .  0.          . -1.35000e+01)

  Surface Number     4      box
    center of box is at x',y',z' coordinate center
    length in x', y', and z' directions       4.00000e+00  4.00000e+00  3.30000e+01
    T/R data
      center of (x',y',z') system   (-2.80000e+01, -1.60000e-01, -3.15000e+01)

  Surface Number     5      box
    center of box is at x',y',z' coordinate center
    length in x', y', and z' directions       4.00000e+00  4.00000e+00  3.30000e+01
    T/R data
      center of (x',y',z') system   (-2.80000e+01,  1.60000e+01, -3.15000e+01)

  Surface Number     6      box
    center of box is at x',y',z' coordinate center
    length in x', y', and z' directions       4.00000e+00  4.00000e+00  3.30000e+01
    T/R data
      center of (x',y',z') system   ( 2.80000e+01,  1.60000e+01, -3.15000e+01)
```

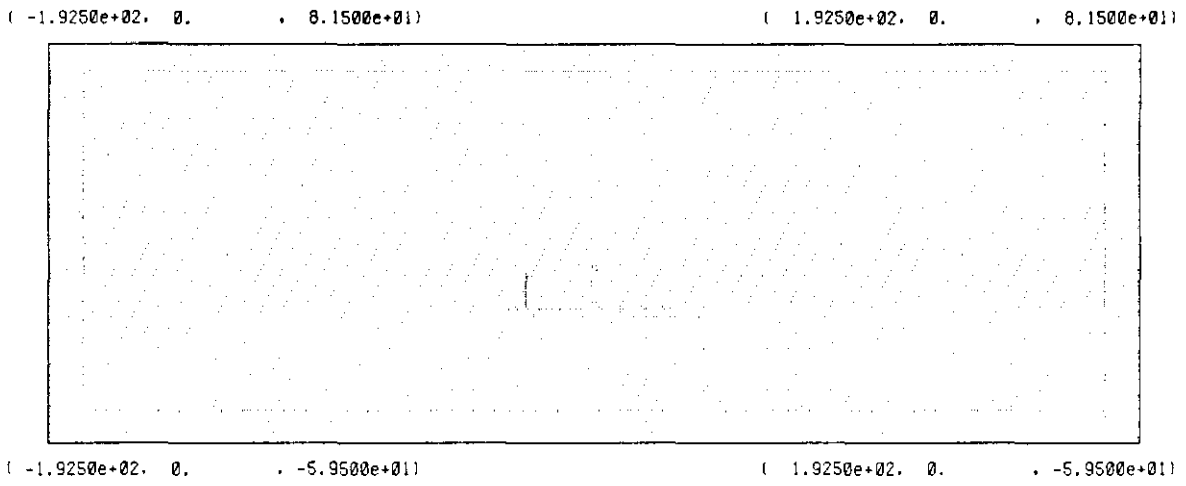
An Example Problem

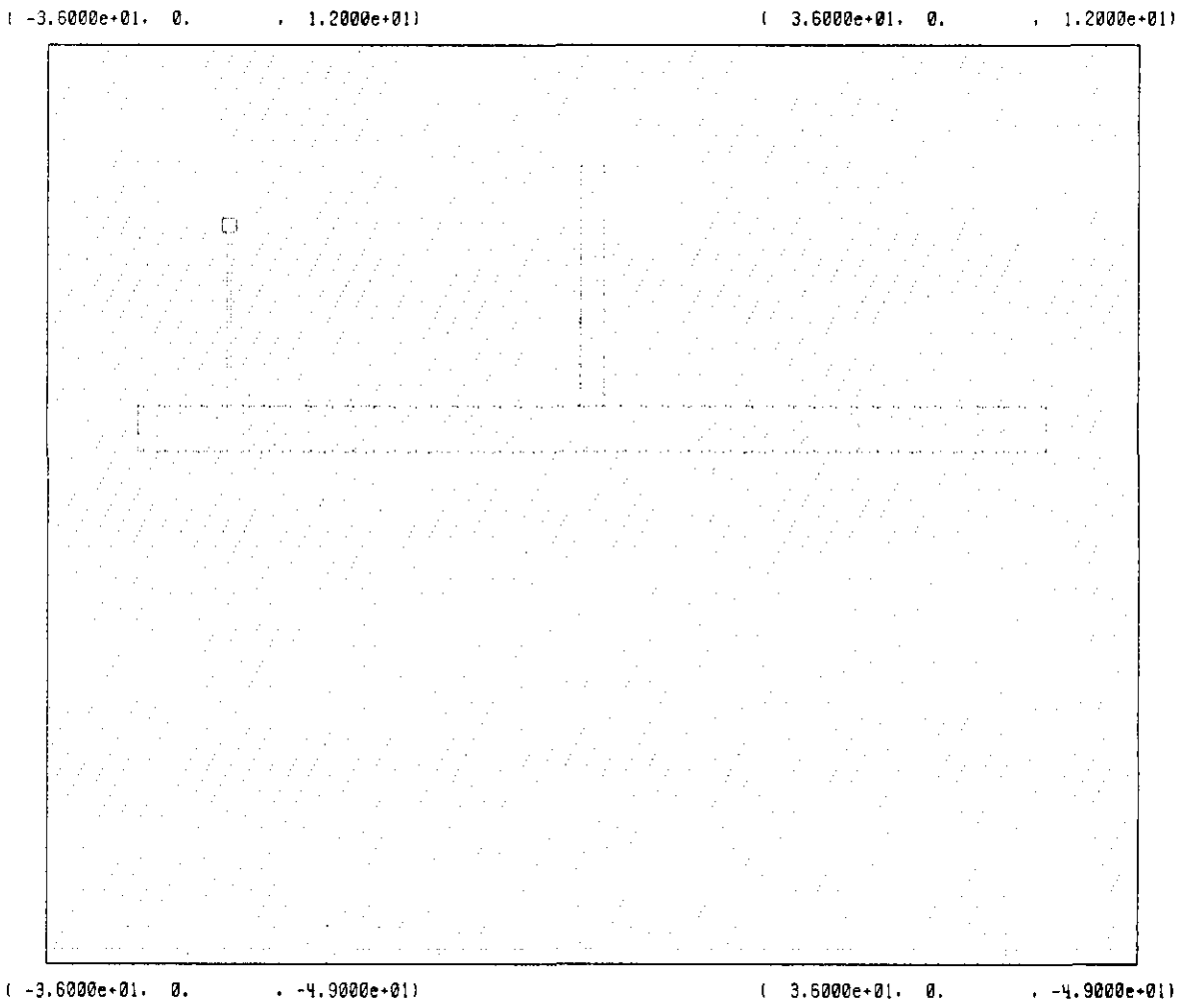
```
Surface Number     7      box
  center of box is at x',y',z' coordinate center
  length in x', y', and z' directions        4.00000e+00  4.00000e+00  3.30000e+01
  T/R data
    center of (x',y',z') system    ( 2.80000e+01, -1.60000e+01, -3.15000e+01)

Surface Number     8      right circular cylinder
  Radius           1.25000e-01
  Surface bounded by planes at x' =  5.00000e-01 and x' =  1.20000e+01
  T/R data
    center of (x',y',z') system    (-2.40000e+01,  0.          ,  0.          )
    point on x'-axis               (-2.40000e+01,  0.          , -1.20000e+02)

Surface Number     9      right circular cylinder
  Radius           5.00000e-01
  Surface bounded by planes at x' = -5.00000e-01 and x' =  5.00000e-01
  T/R data
    center of (x',y',z') system    (-2.40000e+01,  0.          ,  0.          )
    point on x'-axis               (-2.40000e+01,  0.          , -1.20000e+02)

Surface Number    10      right circular cylinder
  Radius           4.37500e-01
  Surface bounded by planes at x' = -4.37500e-01 and x' =  4.37500e-01
  T/R data
    center of (x',y',z') system    (-2.40000e+01,  0.          ,  0.          )
    point on x'-axis               (-2.40000e+01,  0.          , -1.20000e+02)

Surface Number    11      box
  center of box is at x',y',z' coordinate center
  length in x', y', and z' directions        1.50000e+00  8.00000e+00  1.60000e+01
  T/R data
    center of (x',y',z') system    ( 0.          ,  0.          , -4.00000e+00)
```

BASIC GEOMETRICAL DESCRIPTION

	no.	name	med	reg	df	--surface relationships--	
sector	1	walls	1	1	1.000e+00	-1	+2
sector	2	table	2	2	1.000e+00	-3	
					or	-4	
					or	-5	
					or	-6	
					or	-7	
sector	3	holder	3	3	1.000e+00	-8	
					or	-9	+10
sector	4	cobalt	4	4	1.000e+00	-10	
sector	5	lead	5	5	1.000e+00	-11	
boundary		vacuum				+1	
fill		6	6	6	1.000e+00		

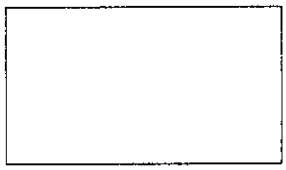
An Example Problem



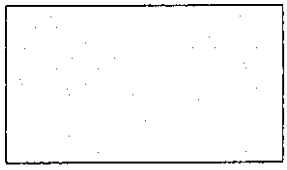

(-3.6000e+01. 0. , 1.2000e+01)
(3.6000e+01. 0. , 1.2000e+01)
(-3.6000e+01. 0. , -4.9000e+01)
(3.6000e+01. 0. , -4.9000e+01)

An Example Problem

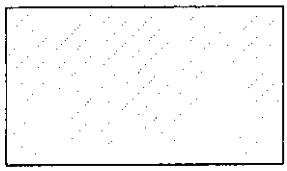
media 0



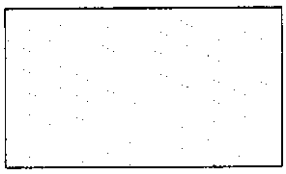
media 1



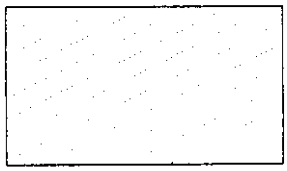
media 2



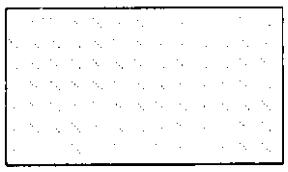
media 3



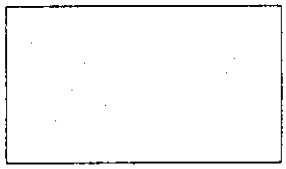
media 4



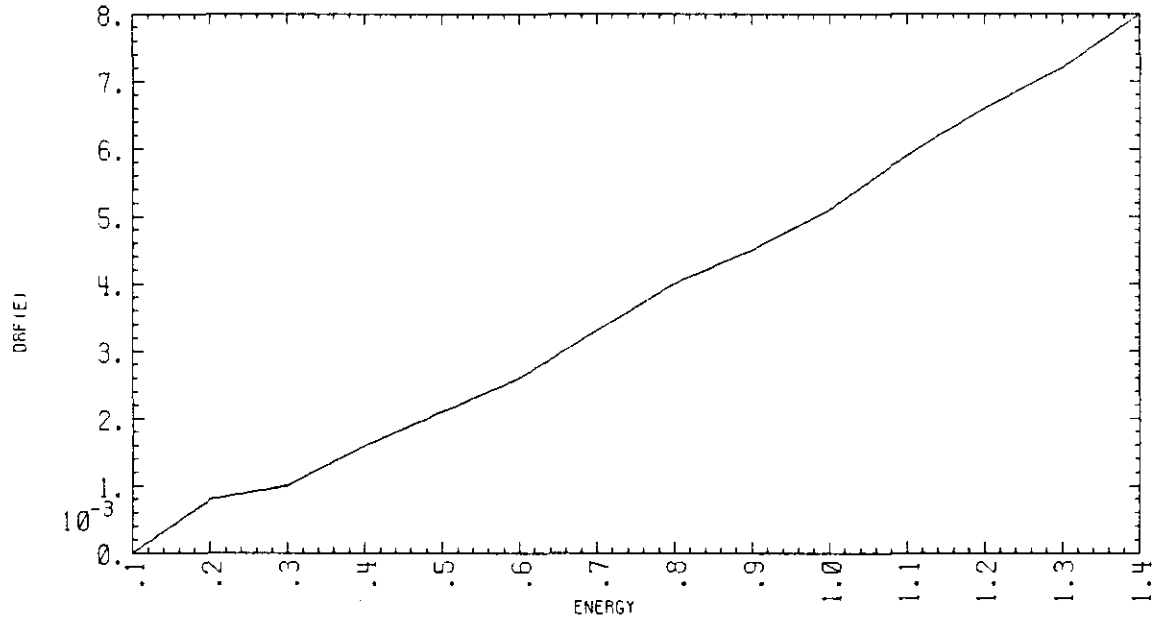
media 5



media 6



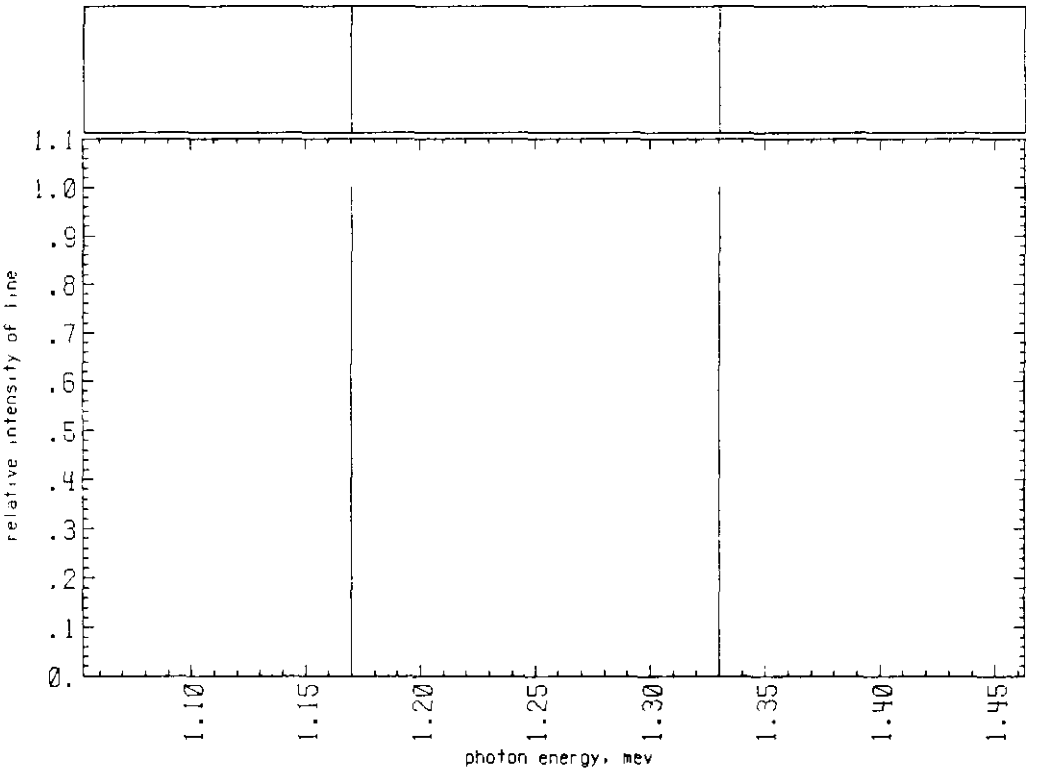
```
Definitions of DETECTORS to be used with this problem

    detector identification number =        1

      type - POINT
      position -  x =  2.400000e+01
                  y =  0.
                  z =  0.           (in units of a in)
      energy dependent detector response function
         photon
         energy input in units of a mev
         (E= 1.0000e-01, DRF= 0.         )   (E= 2.0000e-01, DRF= 8.0000e-04)
         (E= 3.0000e-01, DRF= 1.0000e-03)   (E= 4.0000e-01, DRF= 1.6000e-03)
         (E= 5.0000e-01, DRF= 2.1000e-03)   (E= 6.0000e-01, DRF= 2.6000e-03)
         (E= 7.0000e-01, DRF= 3.3000e-03)   (E= 8.0000e-01, DRF= 4.0000e-03)
         (E= 9.0000e-01, DRF= 4.5000e-03)   (E= 1.0000e+00, DRF= 5.1000e-03)
         (E= 1.1000e+00, DRF= 5.9000e-03)   (E= 1.2000e+00, DRF= 6.6000e-03)
         (E= 1.3000e+00, DRF= 7.2000e-03)   (E= 1.4000e+00, DRF= 8.0000e-03)
```



An Example Problem

```
SOURCE--ENERGY: Definition of description identified by the number    1
    the unit of energy is one mev

    type of particle to be generated = photon

    discrete lines

          energy          intensity

       1.17000e+00     1.00000e+02
       1.33000e+00     1.00000e+00
```

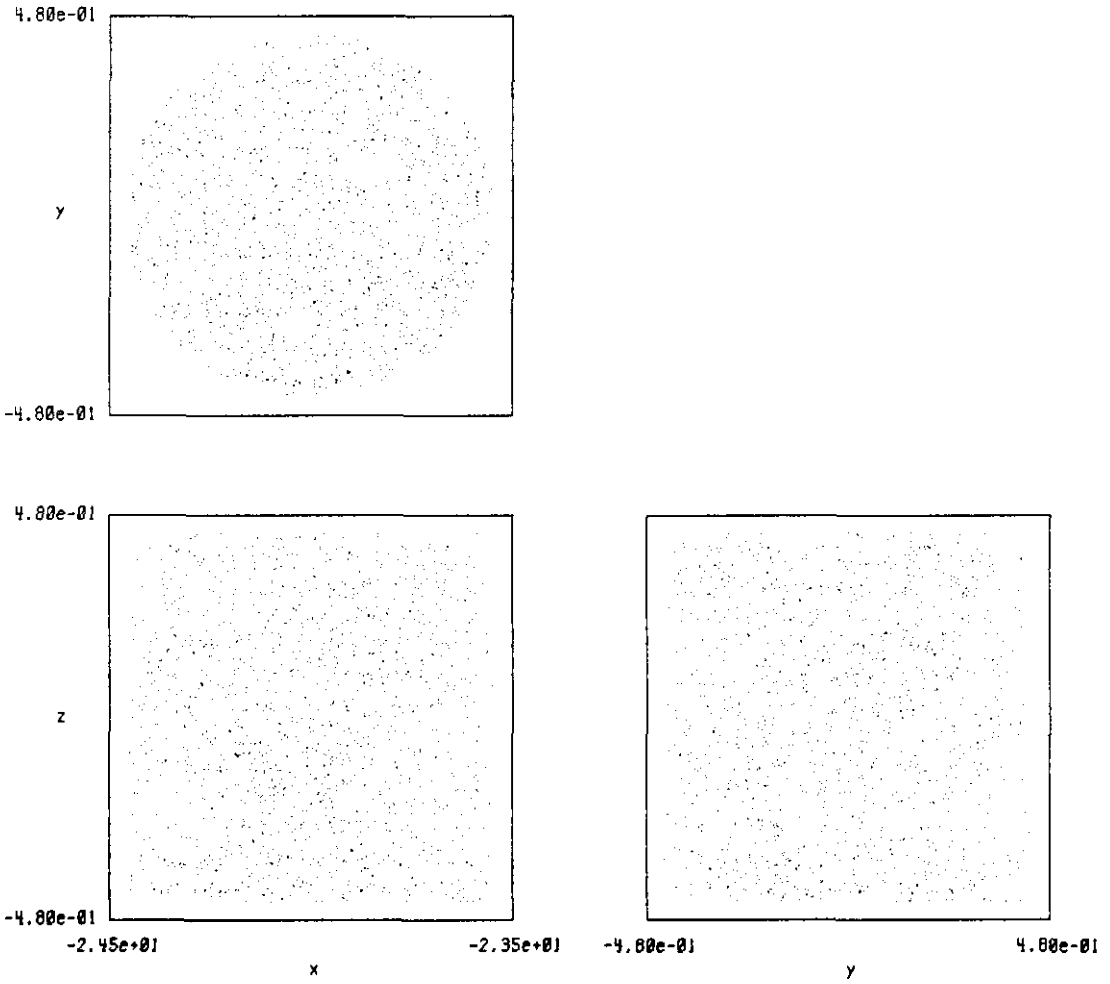
representation of this distribution (including 2000 randomly chosen points)



An Example Problem

```
SOURCE--POSITION: Definition of description identified by the number    1
    the unit of length is one inch

    uniform within a finite cylinder defined by:
      point at the center of the base
                x1=-2.4000e+01   y1= 0.           z1=-4.3750e-01
      point at the center of the top
                x2=-2.4000e+01   y2= 0.           z2= 4.3750e-01
      outer radius = 4.3750e-01
```

location of 2000 points picked at random from this distribution



An Example Problem

```
SOURCE--TIME: Definition of description identified by the number    1
    the unit of time is one second

    steady-state
```

DEFINITION OF FIXED SOURCE

total number of source particles to be generated = 5000

increment	source strength	description of separable parts: position	energy	time	angle	relative importance
1	7.4000e+07 p/s	1	1	1	0	1.0000e+00
totals	7.4000e+07 p/s					

An Example Problem

average source parameters

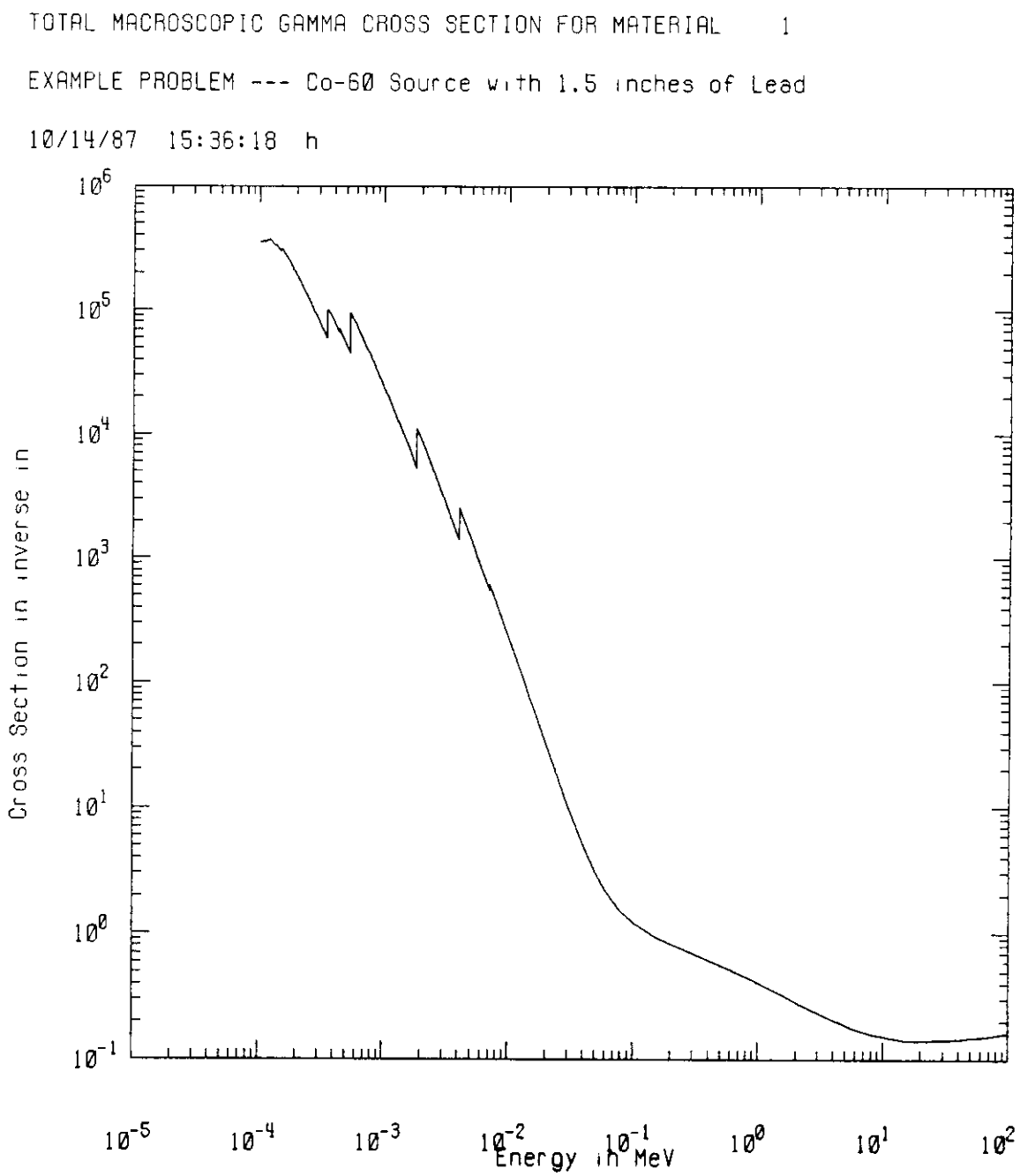
		maximum	minimum	average		
photon						
	x	-2.356e+01	-2.443e+01	-2.400e+01	in	
	y	4.363e-01	-4.360e-01	-3.234e-03	in	
	z	4.373e-01	-4.374e-01	3.609e-03	in	
	u	9.997e-01	-1.000e+00	-4.519e-03		
	v	9.993e-01	-9.996e-01	6.130e-03		
	w	1.000e+00	-9.999e-01	1.816e-02		
	t	0.	0.	0.	sec	
	E	1.330e+00	1.170e+00	1.250e+00		MEV
	wt	1.000e+00	1.000e+00	1.000e+00		

Modifications to the random WALK - splitting/russian roulette at a COLLISION site

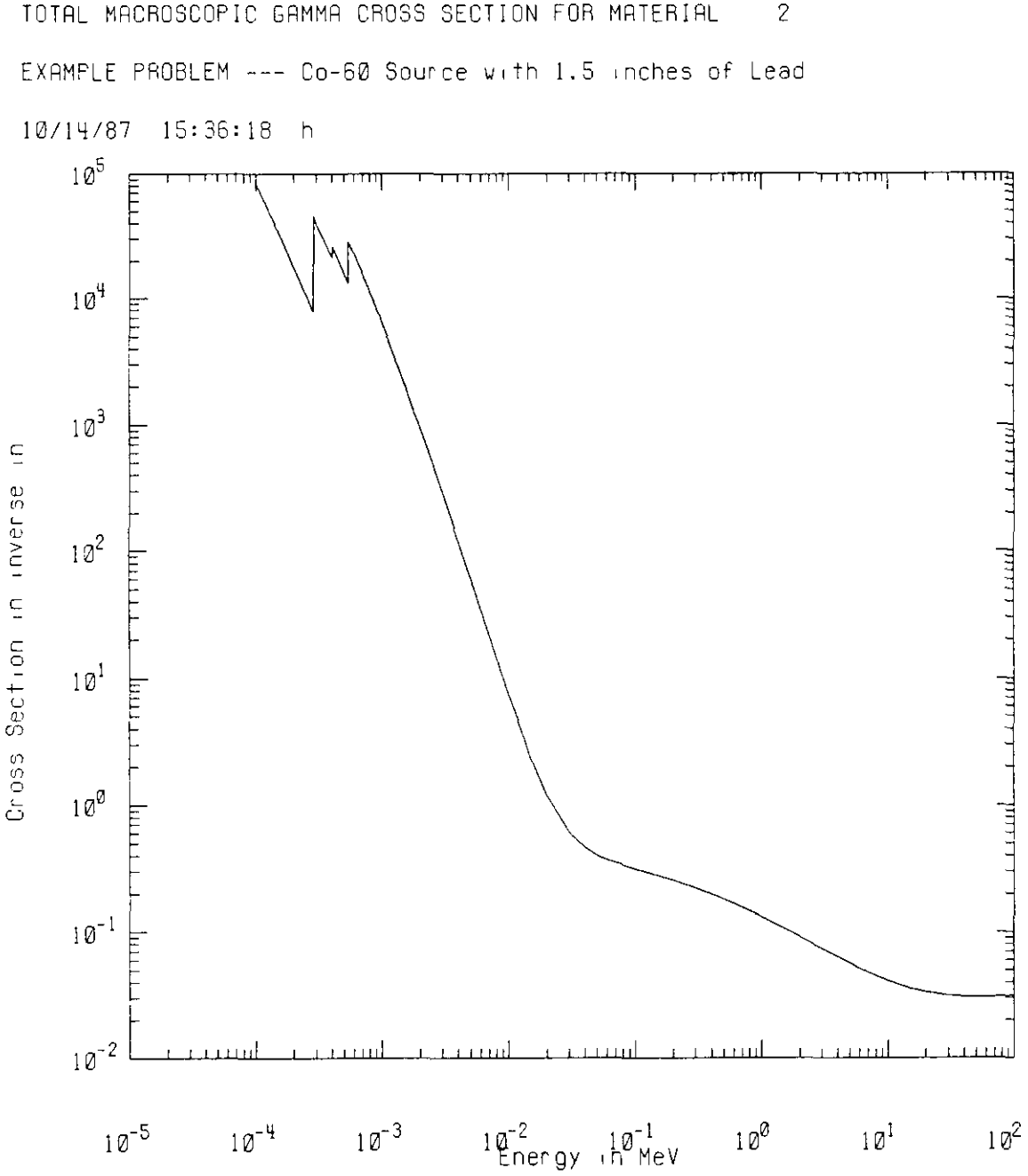
particle type	----regions----	--------energy-------- mev			#-out / #-in
photon	all	0.	to	1.00e-01	0.

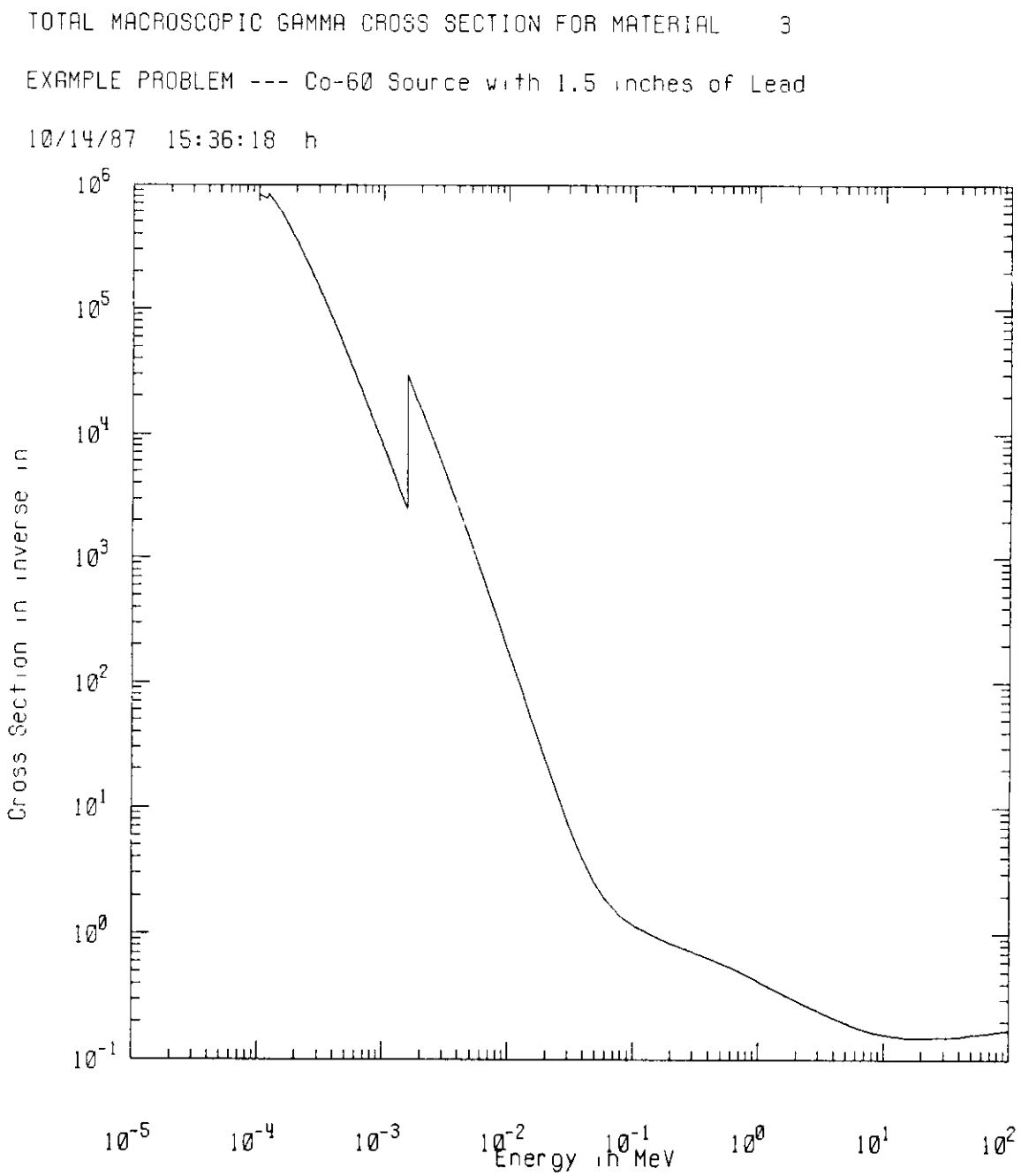
An Example Problem

```
dictionary file ..............   coglex    6/ 9/87

gamma cross section file .....   coggxs    4/ 8/83
```

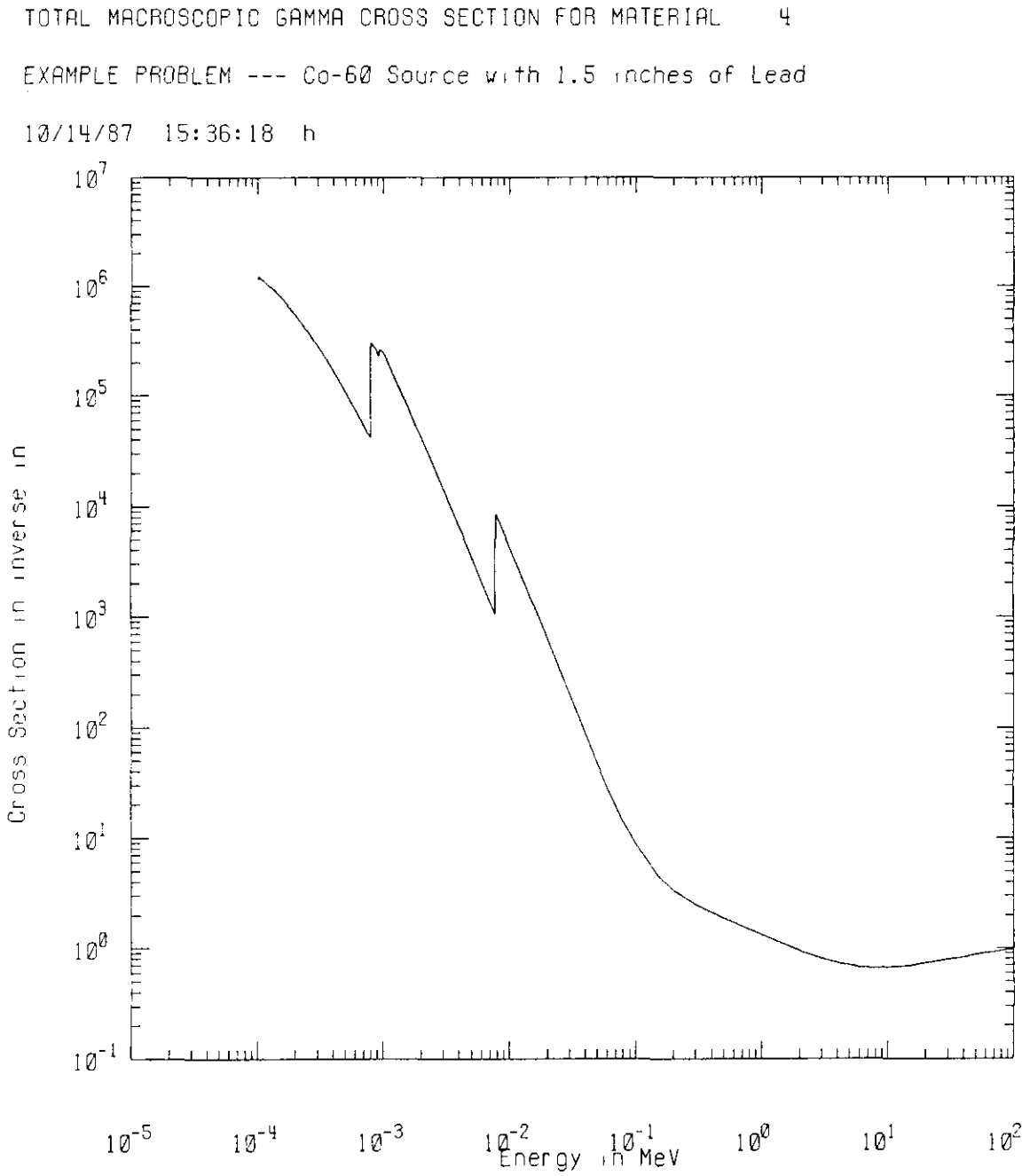

TOTAL MACROSCOPIC GAMMA CROSS SECTION FOR MATERIAL 1
EXAMPLE PROBLEM --- Co-60 Source with 1.5 inches of Lead
10/14/87 15:36:18 h
Cross Section in inverse in
Energy in MeV

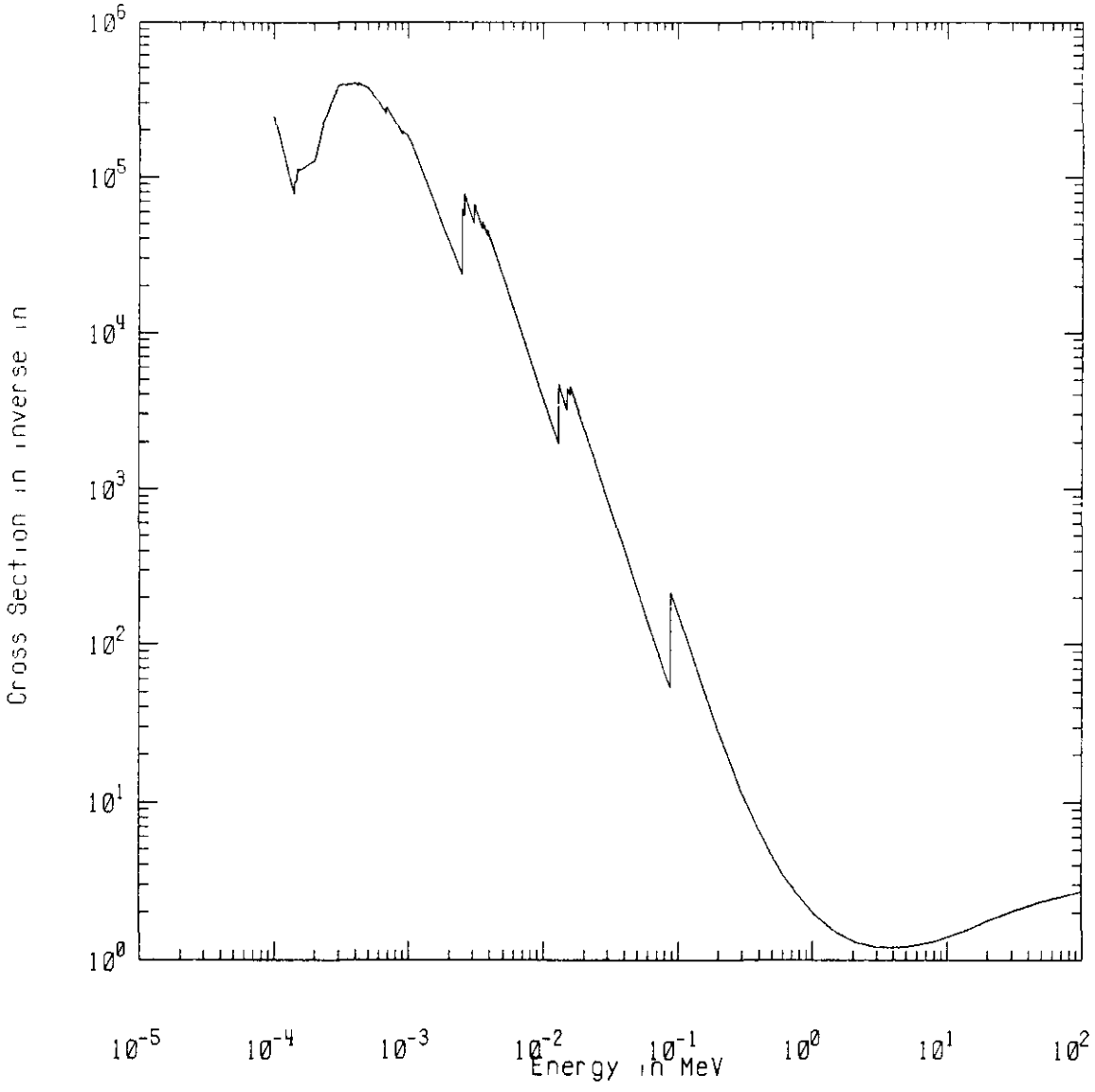
An Example Problem



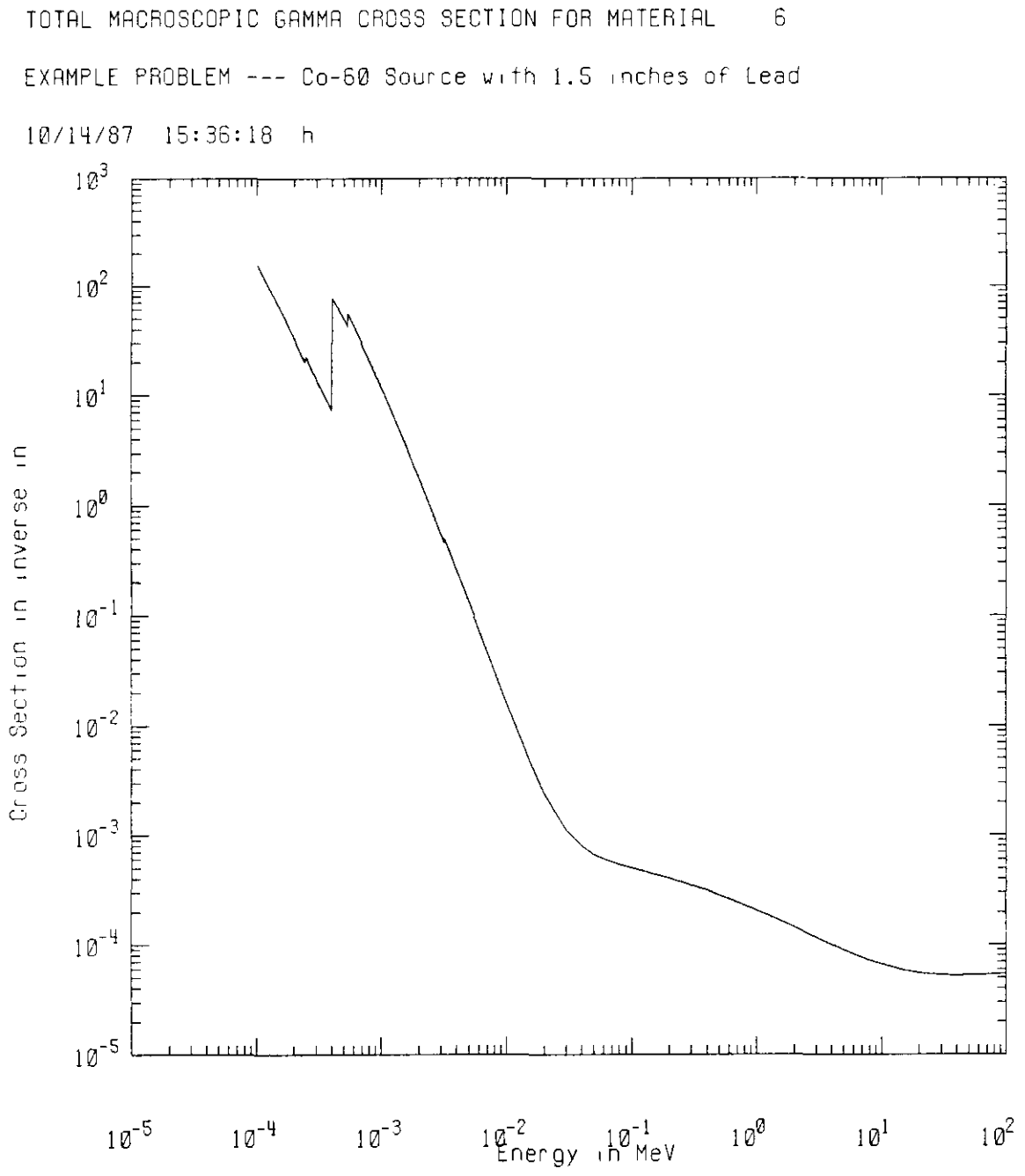

TOTAL MACROSCOPIC GAMMA CROSS SECTION FOR MATERIAL 3
EXAMPLE PROBLEM --- Co-60 Source with 1.5 inches of Lead
10/14/87 15:36:18 h
Cross Section in inverse in
10^6
10^5
10^4
10^3
10^2
10^1
10^0
10^-1
10^-5
10^-4
10^-3
10^-2
10^-1
10^0
10^1
10^2
Energy in MeV

An Example Problem




TOTAL MACROSCOPIC GAMMA CROSS SECTION FOR MATERIAL 5
EXAMPLE PROBLEM --- Co-60 Source with 1.5 inches of Lead
10/14/87 15:36:18 h
Cross Section in inverse in
Energy in MeV

An Example Problem



SUMMARY OF RANDOM-WALK EVENTS FOR 5000 SOURCE PARTICLES

Region number 0

event	particle	# scoring	removal	addition	deposited E.

Region number 1

event	particle	# scoring	removal	addition	deposited E.
current-in	photon	6036	0.	8.9333e+07	0.
current-out	photon	1371	2.0291e+07	0.	0.
leakage	photon	113	1.6724e+06	0.	1.0220e+06
russ. roulette	photon	4381	5.1156e+07	0.	4.3327e+06
rayleigh	photon	818	1.2106e+07	1.2106e+07	0.
compton	photon	26207	3.8786e+08	3.8786e+08	6.0602e+07
photoelectric	photon	1287	1.9048e+07	2.8342e+06	2.7389e+06
totals	photon		4.9214e+08	4.9214e+08	6.8696e+07

Region number 2

event	particle	# scoring	removal	addition	deposited E.

An Example Problem

current-in	photon	1094	0.	1.6191e+07	0.
current-out	photon	1058	1.5658e+07	0.	0.
russ. roulette	photon	36	5.3280e+05	0.	4.7328e+04
rayleigh	photon	4	5.9200e+04	5.9200e+04	0.
compton	photon	754	1.1159e+07	1.1159e+07	3.2630e+06
totals	photon		2.7410e+07	2.7410e+07	3.3104e+06

Region number 3

event	particle	# scoring	removal	addition	deposited E.
current-in	photon	4879	0.	7.2209e+07	0.
current-out	photon	4874	7.2135e+07	0.	0.
russ. roulette	photon	5	5.9728e+04	0.	5.2904e+03
rayleigh	photon	1	1.4800e+04	1.4800e+04	0.
compton	photon	245	3.6260e+06	3.6260e+06	1.4572e+06
photoelectric	photon	1	1.4800e+04	5.2836e+02	3.2080e+03
totals	photon		7.5851e+07	7.5851e+07	1.4657e+06

Region number 4

event	particle	# scoring	removal	addition	deposited E.

fixed source	photon	5000	0.	7.4000e+07	0.
current-in	photon	50	0.	7.4000e+05	0.
current-out	photon	4839	7.1617e+07	0.	0.
russ. roulette	photon	210	1.4099e+06	0.	5.3349e+04
rayleigh	photon	86	1.2728e+06	1.2728e+06	0.
compton	photon	2673	3.9560e+07	3.9560e+07	1.6620e+07
photoelectric	photon	176	2.6048e+06	8.9189e+05	5.9086e+05
totals	photon		1.1647e+08	1.1647e+08	1.7264e+07

Region number 5

event	particle	# scoring	removal	addition	deposited E.
current-in	photon	81	0.	1.1988e+06	0.
current-out	photon	11	1.6280e+05	0.	0.
russ. roulette	photon	178	1.1602e+06	0.	6.7232e+04
rayleigh	photon	9	1.3320e+05	1.3320e+05	0.
compton	photon	82	1.2136e+06	1.2136e+06	4.4519e+05
photoelectric	photon	69	1.0212e+06	1.1454e+06	4.2586e+05
totals	photon		3.6910e+06	3.6910e+06	9.3828e+05

An Example Problem

Region number 6

event	particle	# scoring	removal	addition	deposited E.
current-in	photon	7261	0.	1.0746e+08	0.
current-out	photon	7248	1.0727e+08	0.	0.
russ. roulette	photon	13	1.9240e+05	0.	1.7333e+04
rayleigh	photon	2	2.9600e+04	2.9600e+04	0.
compton	photon	219	3.2412e+06	3.2412e+06	8.2220e+05
totals	photon		1.1073e+08	1.1073e+08	8.3953e+05

Sum over all regions

event	particle	# scoring	removal	addition	deposited E.
fixed source	photon	5000	0.	7.4000e+07	0.
leakage	photon	113	1.6724e+06	0.	1.0220e+06
russ. roulette	photon	4823	5.4511e+07	0.	4.5232e+06
rayleigh	photon	920	1.3616e+07	1.3616e+07	0.
compton	photon	30180	4.4666e+08	4.4666e+08	8.3210e+07
photoelectric	photon	1533	2.2688e+07	4.8721e+06	3.7589e+06
totals	photon		5.3915e+08	5.3915e+08	9.2514e+07

RESULTS FOR DETECTOR 1
EXAMPLE PROBLEM --- Co-60 Source with 1.5 inches of Lead

TYPE -- point

TOTAL RESPONSE
(Integrated over all energies, times and angles and with all masks and response functions included)

particle	response	std. dev.	fsd	fom
photon	2.6256e-01	1.3532e-02	5.1539e-02	4.3540e+01

SOURCE PARTICLES WHICH MADE THE TEN LARGEST CONTRIBUTIONS TO THE TOTAL RESPONSE

event histories for these may be obtained by re-running with the RETRACE option

source particle #	% total result	accumulative %
1981	3.0510	3.0510
4514	1.8714	4.9224
4717	1.6163	6.5387
2331	1.4174	7.9560
2009	1.2580	9.2140
3964	0.9010	10.1149
4720	0.8771	10.9920
3118	0.8290	11.8210
4351	0.7994	12.6204
3072	0.7176	13.3380

An Example Problem

TOTAL RESPONSE AS A FUNCTION OF SCATTERING EVENTS SINCE SOURCE GENERATION

scatterings	% total result	accumulative %
0	54.9513	54.9513
1	21.3484	76.2998
2	13.5104	89.8102
3	5.4609	95.2711
4	2.7350	98.0061
5	1.1751	99.1811
6	0.5610	99.7422
7	0.1310	99.8731
8	0.0526	99.9257
9	0.0327	99.9584
10	0.0257	99.9841
11	0.0102	99.9943
12	0.0035	99.9978
13	0.0010	99.9988
14	0.0012	100.0000
15	0.0000	100.0000
16	0.0000	100.0000

RESULTS FOR DETECTOR 1 (continued)

SEPARATED BY THE STATISTICAL REGION IN WHICH SCATTERINGS OCCURED

region	source particles	collisions	%response	fsd
4	5000	7935	58.828770	0.008
5	5000	160	18.566988	0.262
2	5000	758	12.819724	0.125
1	5000	28312	7.799178	0.026
6	5000	221	1.699456	0.249
3	5000	247	0.285883	0.294

An Example Problem

```
RUNNING TIME

     INPUT
          basic              1.25e-02
          geometry           3.17e-02
          detector           4.85e-03
          source             7.55e-03
          walk               1.13e-04
          cross-section      4.08e-03
          TOTAL                          6.08e-02

     RANDOM-WALK                         8.65e+00

     OUTPUT                              2.86e-03


TOTAL PROBLEM                            8.71e+00  (minutes)
```

Discussion of Results

We will go through the output starting from the problem output immediately after the code prints the listing of the input statements (page 211). We will point out items of interest.

Basic parameters	Additional information is printed reminding you of the units being used, the running time, and the random number used—even though you did not input such data.
Mixture definitions	Adds the total specific gravity so that big errors can more easily be identified.
Geometric surface specifications and basic geometrical description	The printouts add words so that the user can more easily attach a meaning to quantities. We find that users frequently input quantities that are the correct input for a different code but not for COG. The additional printout aids the user in identifying such cases.
Sector information	When sector information is printed, the media, region, and density factor data are included even though these data have already been output (if it was input) in the ASSIGN block printout. This, again, helps to catch input errors.
Detectors	The detector information again reminds the user of the units being used in the problem.

An Example Problem

Source description

Two increments could have been used in the source description. This would have allowed the two different-energy gamma rays to be separated and would have triggered the code to calculate the importance associated with each. Subsequent problems could have been run more effectively using this information. For this specific problem, this is probably of minor concern.

Note that the source strength is double the decay rates, because two gamma rays are produced per decay.

Summary of random walk events

This summary indicates that all requested source particles have been run. This implies that no time cut-off or manual END statement was employed to terminate the problem run. Nothing happened in the void region (number zero), because there was no void in the problem—the default fill was changed to air.

In all regions, the total removal equals the total addition, thus indicating that things were put into the tables correctly. The total deposited energy is statistically equal to the total generated energy. Most of this energy ends up in the concrete walls—a fact that should not be surprising.

Detector 1 results

The total detector response is 0.263 ± 0.014. This has units of:

(photons/cm**2-sec)((counts/photon)(cm**2)) = counts/sec

Note: The calculated flux is in units of photons per cm**2 even though the input units were in inches.

If each source particle scored the same, we would expect (1/7000)(100) or 0.014 percent of the result to come from each. In reality, the largest scoring particle yielded 3.1 percent of the result and the average of the largest ten equals 1.3 percent each. To run more efficiently, we would like to have more particles such as these. The RETRACE option can give us an idea of what makes these particular particles important. (The example output in the section of this report that discusses the RETRACE option (page 196) is this particle.)

Fifty-five percent of the total result came from uncollided photons. This amount could have been calculated by hand using the usual point-kernel techniques.

Twenty-one percent came from the second scattered photons. The maximum number of collisions was 16, but particles scattering this many times did not contribute much to the total response. These particles are of low energy and did not occur close to the detector. If the problem was run longer, it is possible that a particle could be found with this many collisions which would contribute more.

Do not be mislead by the contributions per scattering generation. There well could be a time or energy or angle bin of significance which has all of its contributions from scatterings which register zero on this table.

An Example Problem

When the point estimation is separated by geometric regions, 59 percent of the detector response comes from scatterings in the cobalt—55 percent is the uncollided contribution). Nineteen percent comes from scattering in the lead, 13 percent from the wood table, 8 percent from the concrete walls, 1.7 percent from air scattering, and 0.3 percent from the aluminum.

If you look at the fractional standard deviations, it is evident that the problem could be rerun with limits set to calculate only scatterings in the lead and wood. With more source particles, more accurate solutions could be obtained for these two regions while the problem would run much faster.

Note that we now know more about what is happening physically than an experimentalist could obtain from the physical measurements. It is evident that we are not just measuring the attenuation coefficient in lead, but we have significant contributions from paths that go around the lead blocks.

Running time

An indication is given of the running time in various parts of the problem. One should not worry about running times per source particle, because this depends on the types of detectors used, the complexity of the geometry, and the energy range considered.

References

1. R. J. Howerton, R. E. Dye, and S. T. Perkins, *Evaluated Nuclear Data Library* , Lawrence Livermore National Laboratory, Livermore, CA, UCRL-50400, Vol. 4, Rev. 1 (1981).

2. R. Kinsey, *Data Formats and Procedures for the Evaluated Nuclear Data File, ENDF*, National Nuclear Data Center, Brookhaven National Laboratory, Upton, NY, BNL-NCS-50496 (1979).

3. E. F. Plechaty and J. R. Kimlinger, *TART-NP: A Coupled Neutron–Photon Monte Carlo Transport Code*. Lawrence Livermore National Laboratory, Livermore, CA, UCRL-50400, Vol. 14 (1976).

4. J. R. Kimlinger and E. F. Plechaty, *TART AND ALICE Input Manual*, Lawrence Livermore National Laboratory, Livermore,CA, UCID-17026, Rev. 2 (1986).

5. LASL Group TD-6, *MCNP—A General Monte Carlo Code for Neutron and Photon Transport*, Los Alamos National Laboratory, Los Alamos, NM, LA-7396-M (1978).

6. D. C. Irving, R. M. Freestone, Jr., and F. B. K. Kam, *O5-R, A General Purpose Monte Carlo Neutron Transport Code*, Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-3622 (1965).

7. C. L. Thompson and E. A. Straker, *O6R – ACTIFK, Monte Carlo Neutron Transport Code*, Oak Ridge National Laboratory, Oak Ridge, TN, CF-69-8-36 (1969).

8. E. A. Straker, P. N. Stevens, D. C. Irving, and V. R. Cain, *The MORSE Code—A Multigroup Neutron and Gamma-Ray Monte Carlo Code*, Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-4585 (1970)

9. T. P. Wilcox, *MORSE-L, A Special Version of the MORSE Program Designed to Solve Neutron, Gamma, and Coupled Neutron–Gamma Penetration Problems*, Lawrence Livermore National Laboratory, Livermore, CA, UCID-16680 (1972).

References

10. A. Baur, L. Bourdet, G. deJonghe, A. Monnier, J. C. Nimal, and T. Vergnaud, *TRIPOLI 2, Three Dimensional Polyenergetic Monte Carlo Radiation Transport Program*, CEA/CEN/Saclay SERMA Shielding Laboratory, France, OLS-80-110 (1980).

11. M. O. Cohen, W. Guber, E. Troubetzkoy, H. Lichtenstein, H. Steinbery, and M. Beer, *SAM-CE: A Three Dimensional Monte Carlo Code for the Solution of the Forward Neutron and Forward and Adjoint Gamma Ray Transport Equations—Revision B*, Mathematical Applications Group, Inc., Elmsford, NY,DNA-2830F-B (1973).

12. P. N. Stevens and D. K. Trubey, "Methods for Calculating Neutron and Gamma–Ray Attenuation," in *Weapons Radiation Shielding Handbook*, Oak Ridge National Laboratory, Oak Ridge, TN, DASA-1892-3 (1968), Chapter 3.

13. N. M. Schaeffer, Ed., *Reactor Shielding for Nuclear Engineers*, Radiation Research Associates, Inc., Fort Worth, TX, TID-25951 (1973).

14. H. Goldstein, *Fundamental Aspects of Reactor Shielding* (Addison-Wesley, Reading, MA ,1959).

15. R. E. Maerker and F. J. Muckenthaler, *Calculations, Using the Albedo Concept, of Thermal-Neutron Fluxes, Epi-cadmium Neutron Fluxes, and Fast-Neutron Dose Rates Along the Center Lines of Square Concrete Ducts: Comparison with Experiment*, Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-4147 (1967).

16. G. E. Whitesides and J. T. Thomas, "The Use of Differential Current Albedos in Monte Carlo Criticality Calculations," *Transactions of the American Nuclear Society*, **12**(22), 889 (1969). See, also, **22**, 298 (1975).

17. D. C. Ward, *Monte Carlo Calculations of Light Diffusion Study*, Lawrence Livermore National Laboratory, Livermore, CA, DAG-71-80 (1971).

18. R. Y. Rubinstein, *Simulation and the Monte Carlo Method* (John Wiley & Sons, NY, 1981).

19. L. L. Carter and E. D. Cashwell, *Particle–Transport Simulation with the Monte Carlo Method*, Los Alamos National Laboratory, Los Alamos, NM, TID-26607 (1975).

20. M. H. Kalos, H. Steinberg, and E. Troubetzkey, *Automatic Computation of Importance Sampling Functions for Monte Carlo Transport Codes—Phase III*, Mathematical Applications Group, Inc., Elmsford, NY, DNA-2890F (1972).

21. J. L. Macdonald, *Investigation of Pattern Recognition Techniques for the Identification of Splitting Surfaces in Monte Carlo Particle Trnsport Calculations*, Los Alamos National Laboratory, Los Alamos, NM, LA-6015-T (1975).

22. R. R. Coveyou, V. R. Cain, and K. J. Yost, *Adjoint and Importance In Monte Carlo Application*, Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-4093 (1967).

23. F. H. Clark, *The Exponential Transform as an Importance-Sampling Device—A Review*, Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-RSIC-21 (1968).

24. D. K. Trubey and M. B. Emmett, *A Comparison of First- and Last–Flight Expectation Values Used in an O5R Monte Carlo Calculation of Neutron Distributions in Water*, Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-RSIC-3 (1965).

25. M. A. Gardner and R. J. Howerton, *ACTL: Evaluated Neutron Activation Cross Section Library—Evaluation Techniques and Reaction Index*, Lawrence Livermore National Laboratory, Livermore, CA, UCRL-50400, vol 18 (1978).

26. T. P. Wilcox, "A Method to Calculate Induced Radioactivity Using Existing Transport Codes," *Transactions of the American Nuclear Society*, **22**, 816 (1975).

27. W. E. Selph, *Neutron and Gamma–Ray Albedos*, Oak Ridge National Laboratory, Oak Ridge, TN, ORNL-RSIC-21 (1968).

28. E. Bleuler and G. J. Goldsmith, *Experimental Nucleonics* (Rinehart & Co., Inc., NY, 1952), pp 182-187.

References

Appendix A. Cross-Section Availability and Mixture Definitions

The output of the code that generates the contents of the *dictionary* associated with the COG cross-section libraries is reproduced on the following pages. Under the heading *material* are the allowable names of the elements, isotopes, or mixtures that are defined by the data given to the right. In general, there are a number of names for each entry to accommodate the tastes of users who are used to different naming schemes in the codes they are familiar with.

Note that only the leading eight characters are printed. Entries under `neutron` or `gamma` give the definition of the material in terms of isotopes or elements existing on the neutron or on the gamma cross-section libraries. If no entry is given in one of these two columns, the listed material does not have cross-section data for that particular particle type and, consequently, can not be used.

Elements and isotopes are listed first and are in order of increasing atomic number. These are then followed by mixture definitions. The mixtures should be self-explanatory except for the following:

- `FF` or `99125` represents a composite neutron cross-section for fission fragments.
- `c*3` and `c*9` are simulations of isotropic scattering data with scattering to total cross-section ratios of 0.3 and 0.9. These data do not exist on the standard neutron library. They were used for calculating reference benchmark problems.
- `abs*e`, `c*e`; and `res*abs` are special cross-section sets used for debugging purposes. They are on a special library with `c*3` and `c*9`.

Note: the print out of the `dictionary` indicates its own page numbers. These page numbers are found on the top right-hand corner of each `dictionary` page. These are in addition to the numbers given in this manual. Starting on *dictionary page 31* (this manual, page 283), an index is provided that points to the location of each material, element, and isotope available in the current COG cross-section libraries.

Appendix A. Cross Section Availability and Mixture Definitions

If there are mixtures you regularly use that are not included in the dictionary, the COG programming group would be glad to add them.

coglex 6/9/87 716 entries coglex Page 1

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
h h1 1 1000 1001 hydrogen	8.99e-05	1001	1.00e+00	1000	1.00e+00
d h2 1002 deuteriu	1.80e-04	1002	1.00e+00	1000	1.00e+00
t h3 1003 tritium	2.70e-04	1003	1.00e+00	1000	1.00e+00
he3 2003 helium3	2.70e-04	2003	1.00e+00	2000	1.00e+00
he he4 2 2000 2004 helium helium4	1.78e-04	2004	1.00e+00	2000	1.00e+00
li6 3006 lithium6	4.62e-01	3006	1.00e+00	3000	1.00e+00
li7 3007 lithium7	5.39e-01	3007	1.00e+00	3000	1.00e+00

coglex	6/9/87	716 entries		coglex	Page 2
material	density	neutron isotope	wght frac	gamma isotope	wght frac
li 3 3000 lithium	5.34e-01	3006 3007	6.40e-02 9.36e-01	3000	1.00e+00
be7 4007 berylli7	1.44e+00	4007	1.00e+00	4000	1.00e+00
be be9 4 4000 4009 berylliu be-metal	1.85e+00	4009	1.00e+00	4000	1.00e+00
b10 5010 boron10	2.19e+00	5010	1.00e+00	5000	1.00e+00
b11 5011 boron11	2.41e+00	5011	1.00e+00	5000	1.00e+00
b 5 5000 boron	2.37e+00	5010 5011	1.83e-01 8.17e-01	5000	1.00e+00
c12 6012 carbon12	2.25e+00	6012	1.00e+00	6000	1.00e+00

coglex 6/9/87		716 entries		coglex Page 3	
material	density	neutron isotope	wght frac	gamma isotope	wght frac
c13 6013 carbon13	2.25e+00	6013	1.00e+00	6000	1.00e+00
c 6 6000 carbon graphite	2.25e+00	6012 6013	9.89e-01 1.11e-02	6000	1.00e+00
n14 7014 nitrog14	1.25e-03	7014	1.00e+00	7000	1.00e+00
n15 7015 nitrog15	1.25e-03	7015	1.00e+00	7000	1.00e+00
n 7 7000 nitrogen	1.25e-03	7014 7015	9.96e-01 4.00e-03	7000	1.00e+00
o o16 8 8000 8016 oxygen	1.43e-03	8016	1.00e+00	8000	1.00e+00
f f19 9 9000 9019 fluorine	1.14e+00	9019	1.00e+00	9000	1.00e+00

coglex 6/9/87	716 entries			coglex Page 4	
material density	neutron isotope	wght frac		gamma isotope	wght frac
ne 10 10000 neon	9.00e-04			10000	1.00e+00
na na23 11 11000 11023 sodium	9.71e-01	11023	1.00e+00	11000	1.00e+00
mg 12 12000 magnesiu	1.74e+00	12000	1.00e+00	12000	1.00e+00
al al27 13 13000 13027 aluminum	2.70e+00	13027	1.00e+00	13000	1.00e+00
si 14 14000 silicon	2.33e+00	14000	1.00e+00	14000	1.00e+00
p p31 15 15000 15031 phosphor	1.83e+00	15031	1.00e+00	15000	1.00e+00

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
s32 16032 sulfur32	2.06e+00	16032	1.00e+00	16000	1.00e+00
s 16 16000 sulfur	2.07e+00			16000	1.00e+00
cl 17 17000 chlorine	3.21e-03	17000	1.00e+00	17000	1.00e+00
ar 18 18000 argon a	1.63e-03	18000	1.00e+00	18000	1.00e+00
k 19 19000 potassiu	8.60e-01	19000	1.00e+00	19000	1.00e+00
ca 20 20000 calcium	1.55e+00	20000	1.00e+00	20000	1.00e+00
sc 21 21000 scandium	2.99e+00			21000	1.00e+00

coglex 6/9/87 716 entries coglex Page 6

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
ti 22 22000 titanium	4.51e+00	22000	1.00e+00	22000	1.00e+00
v51 23051 vanadi51	6.10e+00	23051	1.00e+00	23000	1.00e+00
v 23 23000 vanadium	6.10e+00			23000	1.00e+00
cr 24 24000 chromium	7.19e+00	24000	1.00e+00	24000	1.00e+00
mn mn55 25 25000 25055 manganes	7.43e+00	25055	1.00e+00	25000	1.00e+00
fe 26 26000 iron	7.87e+00	26000	1.00e+00	26000	1.00e+00
co co59 27 27000 27059 cobalt	8.85e+00	27059	1.00e+00	27000	1.00e+00

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
ni 28 28000 nickel	8.90e+00	28000	1.00e+00	28000	1.00e+00
ni58 28058 nickel58	8.78e+00	28058	1.00e+00	28000	1.00e+00
cu 29 29000 copper	8.96e+00	29000	1.00e+00	29000	1.00e+00
zn 30 30000 zinc	7.13e+00			30000	1.00e+00
ga 31 31000 galium	5.91e+00	31000	1.00e+00	31000	1.00e+00
ge 32 32000 germaniu	5.32e+00			32000	1.00e+00
as74 33074 arseni74	5.72e+00	33074	1.00e+00	33000	1.00e+00

coglex 6/9/87	716 entries			coglex Page 8	
material density	neutron isotope	wght frac		gamma isotope	wght frac
as	5.72e+00	33075	1.00e+00	33000	1.00e+00
as75					
33					
33000					
33075					
arsenic					
se	4.79e+00			34000	1.00e+00
34					
34000					
selenium					
br	3.12e+00			35000	1.00e+00
35					
35000					
bromine					
kr	3.74e-03			36000	1.00e+00
36					
36000					
krypton					
rb	1.53e+00			37000	1.00e+00
37					
37000					
rubidium					
sr	2.60e+00			38000	1.00e+00
38					
38000					
strontiu					
y88	4.47e+00	39088	1.00e+00	39000	1.00e+00
39088					
yttriu88					

coglex 6/9/87	716 entries			coglex Page 9	
material	density	neutron isotope	wght frac	gamma isotope	wght frac
y y89 39 39000 39089 yttrium	4.47e+00	39089	1.00e+00	39000	1.00e+00
zr 40 40000 zirconiu	6.49e+00	40000	1.00e+00	40000	1.00e+00
nb nb93 41 41000 41093 niobium	8.57e+00	41093	1.00e+00	41000	1.00e+00
mo 42 42000 molybden	1.02e+01	42000	1.00e+00	42000	1.00e+00
tc 43 43000 techneti	1.15e+01			43000	1.00e+00
ru 44 44000 rutheniu	1.22e+01			44000	1.00e+00

coglex 6/9/87	716 entries		coglex Page 10	
material density	neutron isotope	wght frac	gamma isotope	wght frac
rh 45 45000 rhodium	1.24e+01		45000	1.00e+00
pd 46 46000 palladiu	1.20e+01		46000	1.00e+00
ag107 47107 silve107	1.04e+01	47107 1.00e+00	47000	1.00e+00
ag109 47109 silve109	1.06e+01	47109 1.00e+00	47000	1.00e+00
ag 47 47000 silver	1.05e+01	47107 5.14e-01 47109 4.86e-01	47000	1.00e+00
cd 48 48000 cadmium	8.65e+00	48000 1.00e+00	48000	1.00e+00
in 49 49000 indium	7.31e+00		49000	1.00e+00

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
sn 50 50000 tin	7.30e+00	50000	1.00e+00	50000	1.00e+00
sb 51 51000 antimony	6.62e+00			51000	1.00e+00
te 52 52000 telluriu	6.24e+00			52000	1.00e+00
i127 53127 iodin127	4.94e+00	53127	1.00e+00	53000	1.00e+00
i 53 53000 iodine	4.94e+00	53127	1.00e+00	53000	1.00e+00
xe134 54134 xenon134	5.90e-03	54134	1.00e+00	54000	1.00e+00
xe 54 54000 xenon	5.90e-03	54000	1.00e+00	54000	1.00e+00

Appendix A. Cross Section Availability and Mixture Definitions

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
cs 55 55000 cesium	1.90e+00			55000	1.00e+00
ba138 56138 bariu138	3.52e+00	56138	1.00e+00	56000	1.00e+00
ba 56 56000 barium	3.50e+00			56000	1.00e+00
la 57 57000 lanthanu	6.19e+00			57000	1.00e+00
ce 58 58000 cerium	6.77e+00			58000	1.00e+00
pr 59 59000 praseody	6.77e+00			59000	1.00e+00
nd 60 60000 neodymiu	7.00e+00			60000	1.00e+00

coglex 6/9/87		716 entries		coglex Page 13	
material	density	neutron isotope	wght frac	gamma isotope	wght frac
pm 61 61000 promethi	7.10e+00			61000	1.00e+00
sm 62 62000 samarium	7.49e+00			62000	1.00e+00
eu 63 63000 europium	5.25e+00	63000	1.00e+00	63000	1.00e+00
gd 64 64000 gadolini	7.86e+00	64000	1.00e+00	64000	1.00e+00
tb 65 65000 terbium	8.25e+00			65000	1.00e+00
dy 66 66000 dysprosi	8.55e+00			66000	1.00e+00
ho ho165 67 67000 67165 holmium	8.79e+00	67165	1.00e+00	67000	1.00e+00

coglex 6/9/87	716 entries		coglex Page 14	
material density	neutron isotope	wght frac	gamma isotope	wght frac
er 68 68000 erbium	9.15e+00		68000	1.00e+00
tm 69 69000 thulium	9.31e+00		69000	1.00e+00
yb 70 70000 ytterbiu	6.96e+00		70000	1.00e+00
lu 71 71000 lutetium	9.85e+00		71000	1.00e+00
hf 72 72000 hafnium	1.31e+01	72000 1.00e+00	72000	1.00e+00
ta ta181 73 73000 73181 tantalum	1.66e+01	73181 1.00e+00	73000	1.00e+00
w 74 74000 tungsten	1.93e+01	74000 1.00e+00	74000	1.00e+00

coglex 6/9/87 716 entries coglex Page 15

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
re185 75185 rheni185	2.09e+01	75185	1.00e+00	75000	1.00e+00
re187 75187 rheni187	2.11e+01	75187	1.00e+00	75000	1.00e+00
re 75 75000 rhenium	2.10e+01	75185 75187	3.68e-01 6.32e-01	75000	1.00e+00
os 76 76000 osmium	2.26e+01			76000	1.00e+00
ir 77 77000 iridium	2.25e+01			77000	1.00e+00
pt 78 78000 platinum	2.15e+01	78000	1.00e+00	78000	1.00e+00
au au197 79 79000 79197 gold	1.93e+01	79197	1.00e+00	79000	1.00e+00

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
hg 80 80000 mercury	1.36e+01			80000	1.00e+00
tl 81 81000 thallium	1.19e+01			81000	1.00e+00
pb 82 82000 lead	1.14e+01	82000	1.00e+00	82000	1.00e+00
bi bi209 83 83000 83209 bismuth	9.80e+00	83209	1.00e+00	83000	1.00e+00
po 84 84000 polonium	9.24e+00			84000	1.00e+00
at 85 85000 astatine	1.00e+01			85000	1.00e+00
rn 86 86000 radon	9.60e-03			86000	1.00e+00

coglex 6/9/87 716 entries coglex Page 17

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
fr 87 87000 francium	1.00e+01			87000	1.00e+00
ra 88 88000 radium	5.00e+00			88000	1.00e+00
ac 89 89000 actinium	1.00e+01			89000	1.00e+00
th231 90231 thori231	1.17e+01	90231	1.00e+00	90000	1.00e+00
th233 90233 thori233	1.17e+01	90233	1.00e+00	90000	1.00e+00
th th232 90 90000 90232 thorium	1.17e+01	90232	1.00e+00	90000	1.00e+00
pa233 91233 prota233	1.54e+01	91233	1.00e+00	91000	1.00e+00

coglex 6/9/87 716 entries coglex Page 18

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
pa 91 91000 protacti	1.54e+01			91000	1.00e+00
u233 92233 urani233	1.86e+01	92233	1.00e+00	92000	1.00e+00
u234 92234 urani234	1.87e+01	92234	1.00e+00	92000	1.00e+00
u235 92235 urani235	1.88e+01	92235	1.00e+00	92000	1.00e+00
u236 92236 urani236	1.88e+01	92236	1.00e+00	92000	1.00e+00
u237 92237 urani237	1.89e+01	92237	1.00e+00	92000	1.00e+00
u238 92238 urani238	1.90e+01	92238	1.00e+00	92000	1.00e+00
u239 92239 urani239	1.91e+01	92239	1.00e+00	92000	1.00e+00

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
u240 92240 urani240	1.92e+01	92240	1.00e+00	92000	1.00e+00
u 92 92000 uranium	1.90e+01	92235 92238	7.00e-03 9.93e-01	92000	1.00e+00
np235 93235 neptu235	1.95e+01	93235	1.00e+00	93000	1.00e+00
np236 93236 neptu236	1.95e+01	93236	1.00e+00	93000	1.00e+00
np238 93238 neptu238	1.95e+01	93238	1.00e+00	93000	1.00e+00
np np237 93 93000 93237 neptuniu	1.95e+01	93237	1.00e+00	93000	1.00e+00
pu237 94237 pluto237	1.96e+01	94237	1.00e+00	94000	1.00e+00
pu238 94238 pluto238	1.96e+01	94238	1.00e+00	94000	1.00e+00

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
pu239 94239 pluto239	1.96e+01	94239	1.00e+00	94000	1.00e+00
pu240 94240 pluto240	1.96e+01	94240	1.00e+00	94000	1.00e+00
pu241 94241 pluto241	1.96e+01	94241	1.00e+00	94000	1.00e+00
pu243 94243 pluto243	1.96e+01	94243	1.00e+00	94000	1.00e+00
pu pu242 94 94000 94242 plutoniu	1.96e+01	94242	1.00e+00	94000	1.00e+00
am241 95241 ameri241	1.36e+01	95241	1.00e+00	95000	1.00e+00
am242 95242 ameri242	1.36e+01	95242	1.00e+00	95000	1.00e+00
am243 95243 ameri243	1.37e+01	95243	1.00e+00	95000	1.00e+00

coglex 6/9/87	716 entries			coglex Page 21	
material	density	neutron isotope	wght frac	gamma isotope	wght frac
am 95 95000 americiu	1.37e+01			95000	1.00e+00
cm242 96242 curiu242	1.00e+01	96242	1.00e+00	96000	1.00e+00
cm243 96243 curiu243	1.00e+01	96243	1.00e+00	96000	1.00e+00
cm244 96244 curiu244	1.00e+01	96244	1.00e+00	96000	1.00e+00
cm245 96245 curiu245	1.00e+01	96245	1.00e+00	96000	1.00e+00
cm246 96246 curiu246	1.00e+01	96246	1.00e+00	96000	1.00e+00
cm247 96247 curiu247	1.00e+01	96247	1.00e+00	96000	1.00e+00
cm248 96248 curiu248	1.00e+01	96248	1.00e+00	96000	1.00e+00

coglex 6/9/87	716 entries		coglex Page 22	
material density	neutron isotope	wght frac	gamma isotope	wght frac
cm 96 96000 curium 1.00e+01			96000	1.00e+00
bk249 97249 berke249 1.00e+01	97249	1.00e+00	97000	1.00e+00
bk 97 97000 berkeliu 1.00e+01			97000	1.00e+00
cf249 98249 calif249 1.00e+01	98249	1.00e+00	98000	1.00e+00
cf250 98250 calif250 1.00e+01	98250	1.00e+00	98000	1.00e+00
cf251 98251 calif251 1.00e+01	98251	1.00e+00	98000	1.00e+00
cf252 98252 calif252 1.00e+01	98252	1.00e+00	98000	1.00e+00
cf 98 98000 californ 1.00e+01			98000	1.00e+00

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
es 99 99000 einstein	1.00e+01			99000	1.00e+00
ff 99125 fissionf	1.00e+00	99125	1.00e+00		
fm 100 100000 fermium	1.00e+01			100000	1.00e+00
air	1.29e-03	7014 8016 18000	7.55e-01 2.32e-01 1.30e-02	7000 8000 18000	7.55e-01 2.32e-01 1.30e-02
polystyr ch	1.05e+00	1001 6012	7.70e-02 9.23e-01	1000 6000	7.70e-02 9.23e-01
polyethy ch2	9.13e-01	1001 6012	1.44e-01 8.56e-01	1000 6000	1.44e-01 8.56e-01
pva c2h3o	1.17e+00	1001 6012 8016	7.00e-02 5.58e-01 3.72e-01	1000 6000 8000	7.00e-02 5.58e-01 3.72e-01
pvc c2h3cl	1.40e+00	1001 6012 17000	4.80e-02 3.84e-01 5.68e-01	1000 6000 17000	4.80e-02 3.84e-01 5.68e-01

Appendix A. Cross Section Availability and Mixture Definitions

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
teflon cf2	2.17e+00	6012 9019	2.40e-01 7.60e-01	6000 9000	2.40e-01 7.60e-01
paryl-n c8h8	1.11e+00	1001 6012	1.29e-01 8.71e-01	1000 6000	1.29e-01 8.71e-01
paryl-c c4h3cl	1.00e+00	1001 6012 17000	3.50e-02 5.55e-01 4.10e-01	1000 6000 17000	3.50e-02 5.55e-01 4.10e-01
mylar c10h8o4	1.40e+00	1001 6012 8016	4.20e-02 6.25e-01 3.33e-01	1000 6000 8000	4.20e-02 6.25e-01 3.33e-01
saran c2h2cl2	1.68e+00	1001 6012 17000	2.10e-02 2.48e-01 7.31e-01	1000 6000 17000	2.10e-02 2.48e-01 7.31e-01
lucite c5h8o2	1.18e+00	1001 6012 8016	8.00e-02 6.00e-01 3.20e-01	1000 6000 8000	8.00e-02 6.00e-01 3.20e-01
formvar c10h16o5	1.23e+00	1001 6012 8016	7.40e-02 5.58e-01 3.68e-01	1000 6000 8000	7.40e-02 5.58e-01 3.68e-01
polyform ch2o	1.41e+00	1001 6012 8016	6.70e-02 4.00e-01 5.33e-01	1000 6000 8000	6.70e-02 4.00e-01 5.33e-01

coglex	6/9/87	716 entries		coglex	Page 25
material	density	neutron isotope	wght frac	gamma isotope	wght frac
fbrglass	2.54e+00	1001	1.40e-03	1000	1.40e-03
fg		6012	1.59e-01	6000	1.59e-01
		7014	1.36e-02	7000	1.36e-02
		8016	3.43e-01	8000	3.43e-01
		12000	5.20e-02	12000	5.20e-02
		13027	1.27e-01	13000	1.27e-01
		14000	3.04e-01	14000	3.04e-01
beoxide	3.03e+00	4009	3.60e-01	4000	3.60e-01
beo		8016	6.40e-01	8000	6.40e-01
lih	8.20e-01	1001	1.27e-01	1000	1.27e-01
		3006	5.60e-02	3000	8.73e-01
		3007	8.17e-01		
li7h	8.27e-01	1001	1.26e-01	1000	1.26e-01
		3007	8.74e-01	3000	8.74e-01
lif	2.64e+00	3006	1.80e-02	3000	2.68e-01
		3007	2.50e-01	9000	7.32e-01
		9019	7.32e-01		
aloxide	3.70e+00	8016	4.71e-01	8000	4.71e-01
al2o3		13027	5.29e-01	13000	5.29e-01
feoxide	5.20e+00	8016	3.01e-01	8000	3.01e-01
feo		26000	6.99e-01	26000	6.99e-01
sio2	2.64e+00	8016	5.33e-01	8000	5.33e-01
quartz		14000	4.67e-01	14000	4.67e-01
nai	3.67e+00	11023	1.53e-01	11000	1.53e-01
		53127	8.47e-01	53000	8.47e-01

Appendix A. Cross Section Availability and Mixture Definitions

coglex 6/9/87 716 entries coglex Page 26

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
steel	7.78e+00	6012	4.00e-04	6000	4.00e-04
ss304		14000	4.60e-03	14000	4.60e-03
		24000	1.90e-01	24000	1.90e-01
		25055	9.00e-03	25000	9.00e-03
		26000	6.99e-01	26000	6.99e-01
		28000	9.70e-02	28000	9.70e-02
water	1.00e+00	1001	1.12e-01	1000	1.12e-01
ice		8016	8.88e-01	8000	8.88e-01
h2o					
d2o	1.11e+00	1002	2.01e-01	1000	2.01e-01
hvywater		8016	7.99e-01	8000	7.99e-01
concrete	2.25e+00	1001	5.50e-03	1000	5.50e-03
lllcncrt		8016	4.85e-01	8000	4.85e-01
cncrt1		11023	2.57e-02	11000	2.57e-02
		12000	1.50e-02	12000	1.50e-02
		13027	4.29e-02	13000	4.29e-02
		14000	3.31e-01	14000	3.31e-01
		20000	6.01e-02	20000	6.01e-02
		26000	3.41e-02	26000	3.41e-02
orcncrt	2.51e+00	1001	5.00e-03	1000	5.00e-03
cncrt2		8016	4.35e-01	8000	4.35e-01
		11023	1.83e-02	11000	1.83e-02
		12000	1.34e-02	12000	1.34e-02
		13027	6.20e-03	13000	6.20e-03
		14000	3.01e-01	14000	3.01e-01
		20000	1.90e-01	20000	1.90e-01
		26000	3.08e-02	26000	3.08e-02

Appendix A. Cross Section Availability and Mixture Definitions

coglex 6/9/87		716 entries		coglex	Page 28
material	density	neutron isotope	wght frac	gamma isotope	wght frac
benzene *c6h6*	8.79e-01	1001	7.70e-02	1000	7.70e-02
		6012	9.23e-01	6000	9.23e-01
zrh2	5.61e+00	1001	2.20e-02	1000	2.20e-02
		40000	9.78e-01	40000	9.78e-01
uo2	1.00e+01	8016	1.18e-01	8000	1.18e-01
		92235	6.00e-03	92000	8.82e-01
		92238	8.76e-01		
u233o2	1.00e+01	8016	1.21e-01	8000	1.21e-01
		92233	8.79e-01	92000	8.79e-01
u234o2	1.00e+01	8016	1.20e-01	8000	1.20e-01
		92234	8.80e-01	92000	8.80e-01
u235o2	1.00e+01	8016	1.20e-01	8000	1.20e-01
		92235	8.80e-01	92000	8.80e-01
u236o2	1.00e+01	8016	1.19e-01	8000	1.19e-01
		92236	8.81e-01	92000	8.81e-01
u237o2	1.00e+01	8016	1.19e-01	8000	1.19e-01
		92237	8.81e-01	92000	8.81e-01
u238o2	1.00e+01	8016	1.19e-01	8000	1.19e-01
		92238	8.81e-01	92000	8.81e-01
u239o2	1.00e+01	8016	1.18e-01	8000	1.18e-01
		92239	8.82e-01	92000	8.82e-01

coglex	6/9/87	716 entries		coglex	Page 27
material	density	neutron isotope	wght frac	gamma isotope	wght frac
mfecncrt	2.40e+00	1001	3.90e-03	1000	3.90e-03
cncrt3		6012	1.30e-02	6000	1.30e-02
		8016	5.09e-01	8000	5.09e-01
		11023	1.45e-02	11000	1.45e-02
		12000	1.20e-02	12000	1.20e-02
		13027	5.00e-02	13000	5.00e-02
		14000	3.05e-01	14000	3.05e-01
		19000	1.50e-03	19000	1.50e-03
		20000	6.20e-02	20000	6.20e-02
		26000	2.90e-02	26000	2.90e-02
barytes	3.35e+00	1001	4.00e-03	1000	4.00e-03
cncrt4		8016	3.11e-01	8000	3.11e-01
		12000	1.00e-03	12000	1.00e-03
		13027	4.00e-03	13000	4.00e-03
		14000	1.10e-02	14000	1.10e-02
		16032	1.08e-01	16000	1.08e-01
		20000	5.00e-02	20000	5.00e-02
		26000	4.80e-02	26000	4.80e-02
		56138	4.63e-01	56000	4.63e-01
mgnetite	3.52e+00	1001	3.00e-03	1000	3.00e-03
cncrt5		8016	3.30e-01	8000	3.30e-01
		12000	9.00e-03	12000	9.00e-03
		13027	2.40e-02	13000	2.40e-02
		14000	2.60e-02	14000	2.60e-02
		16032	1.00e-03	16000	1.00e-03
		20000	7.10e-02	20000	7.10e-02
		22000	5.50e-02	22000	5.50e-02
		23051	3.00e-03	23000	3.00e-03
		24000	2.00e-03	24000	2.00e-03
		25055	2.00e-03	25000	2.00e-03
		26000	4.73e-01	26000	4.73e-01
ilmenite	3.62e+00	1001	6.00e-03	1000	6.00e-03
cncrt6		8016	3.80e-01	8000	3.80e-01
		12000	7.90e-02	12000	7.90e-02
		13027	6.00e-03	13000	6.00e-03
		14000	2.80e-02	14000	2.80e-02
		20000	6.50e-02	20000	6.50e-02
		22000	2.70e-01	22000	2.70e-01
		25055	7.00e-03	25000	7.00e-03
		26000	1.59e-01	26000	1.59e-01

coglex 6/9/87	716 entries		coglex Page 29	
material density	neutron isotope	wght frac	gamma isotope	wght frac
u240o2 1.00e+01	8016	1.18e-01	8000	1.18e-01
	92240	8.82e-01	92000	8.82e-01
d38 1.89e+01	92235	7.00e-03	92000	1.00e+00
tuballoy	92238	9.93e-01		
oy 1.89e+01	92235	9.32e-01	92000	1.00e+00
oralloy	92238	6.80e-02		
1x040 1.89e+00	1001	2.60e-02	1000	2.60e-02
1x041	6012	1.86e-01	6000	1.86e-01
	7014	3.22e-01	7000	3.22e-01
	8016	3.67e-01	8000	3.67e-01
	9019	9.90e-02	9000	9.90e-02
1x07 1.89e+00	1001	2.60e-02	1000	2.60e-02
	6012	1.78e-01	6000	1.78e-01
	7014	3.40e-01	7000	3.40e-01
	8016	3.89e-01	8000	3.89e-01
	9019	6.70e-02	9000	6.70e-02
1x09 1.89e+00	1001	2.80e-02	1000	2.80e-02
	6012	1.72e-01	6000	1.72e-01
	7014	3.63e-01	7000	3.63e-01
	8016	4.14e-01	8000	4.14e-01
	9019	3.00e-03	9000	3.00e-03
1x10 1.89e+00	1001	2.70e-02	1000	2.70e-02
	6012	1.69e-01	6000	1.69e-01
	7014	3.59e-01	7000	3.59e-01
	8016	4.11e-01	8000	4.11e-01
	9019	3.40e-02	9000	3.40e-02

coglex 6/9/87 716 entries coglex Page 30

material	density	neutron isotope	neutron wght frac	gamma isotope	gamma wght frac
pbx9404 pbx9409	1.89e+00	1001 6012 7014 8016 15031 17000	2.80e-02 1.68e-01 3.60e-01 4.30e-01 3.00e-03 1.10e-02	1000 6000 7000 8000 15000 17000	2.80e-02 1.68e-01 3.60e-01 4.30e-01 3.00e-03 1.10e-02
mockhe mock-he lm-04-0	1.89e+00	1001 6012 7014 8016	3.40e-02 3.13e-01 3.08e-01 3.45e-01	1000 6000 7000 8000	3.40e-02 3.13e-01 3.08e-01 3.45e-01
c*3	1.00e+50	2	1.00e+00		
c*9	1.00e+50	3	1.00e+00		
abs*e	1.00e+50	4	1.00e+00		
c*e	1.00e+50	5	1.00e+00		
res*abs	1.00e+50	6	1.00e+00		

alphabetical listing

material	page	material	page	material	page	material	page
1	1	17	5	26000	6	35	8
10	4	17000	5	27	6	35000	8
100	23	18	5	27000	6	36	8
1000	1	18000	5	27059	6	36000	8
10000	4	19	5	28	7	37	8
100000	23	19000	5	28000	7	37000	8
1001	1	2	1	28058	7	38	8
1002	1	20	5	29	7	38000	8
1003	1	2000	1	29000	7	39	9
11	4	20000	5	3	2	39000	9
11000	4	2003	1	30	7	39088	8
11023	4	2004	1	3000	2	39089	9
12	4	21	5	30000	7	4	2
12000	4	21000	5	3006	1	40	9
13	4	22	6	3007	1	4000	2
13000	4	22000	6	31	7	40000	9
13027	4	23	6	31000	7	4007	2
14	4	23000	6	32	7	4009	2
14000	4	23051	6	32000	7	41	9
15	4	24	6	33	8	41000	9
15000	4	24000	6	33000	8	41093	9
15031	4	25	6	33074	7	42	9
16	5	25000	6	33075	8	42000	9
16000	5	25055	6	34	8	43	9
16032	5	26	6	34000	8	43000	9

coglex 6/9/87 716 entries coglex Page 32

alphabetical listing

material	page	material	page	material	page	material	page
44	9	53000	11	62000	13	72000	14
44000	9	53127	11	63	13	73	14
45	10	54	11	63000	13	73000	14
45000	10	54000	11	64	13	73181	14
46	10	54134	11	64000	13	74	14
46000	10	55	12	65	13	74000	14
47	10	55000	12	65000	13	75	15
47000	10	56	12	66	13	75000	15
47107	10	56000	12	66000	13	75185	15
47109	10	56138	12	67	13	75187	15
48	10	57	12	67000	13	76	15
48000	10	57000	12	67165	13	76000	15
49	10	58	12	68	14	77	15
49000	10	58000	12	68000	14	77000	15
5	2	59	12	69	14	78	15
50	11	59000	12	69000	14	78000	15
5000	2	6	3	7	3	79	15
50000	11	60	12	70	14	79000	15
5010	2	6000	3	7000	3	79197	15
5011	2	60000	12	70000	14	8	3
51	11	6012	2	7014	3	80	16
51000	11	6013	3	7015	3	8000	3
52	11	61	13	71	14	80000	16
52000	11	61000	13	71000	14	8016	3
53	11	62	13	72	14	81	16

coglex 6/9/87 716 entries coglex Page 33

alphabetical listing

material	page	material	page	material	page	material	page
81000	16	90233	17	94240	20	98251	22
82	16	91	18	94241	20	98252	22
82000	16	91000	18	94242	20	99	23
83	16	91233	17	94243	20	99000	23
83000	16	92	19	95	21	99125	23
83209	16	92000	19	95000	21	a	5
84	16	92233	18	95241	20	abs*e	30
84000	16	92234	18	95242	20	ac	17
85	16	92235	18	95243	20	actinium	17
85000	16	92236	18	96	22	ag	10
86	16	92237	18	96000	22	ag107	10
86000	16	92238	18	96242	21	ag109	10
87	17	92239	18	96243	21	air	23
87000	17	92240	19	96244	21	al	4
88	17	93	19	96245	21	al27	4
88000	17	93000	19	96246	21	al2o3	25
89	17	93235	19	96247	21	aloxide	25
89000	17	93236	19	96248	21	aluminum	4
9	3	93237	19	97	22	am	21
90	17	93238	19	97000	22	am241	20
9000	3	94	20	97249	22	am242	20
90000	17	94000	20	98	22	am243	20
9019	3	94237	19	98000	22	ameri241	20
90231	17	94238	19	98249	22	ameri242	20
90232	17	94239	20	98250	22	ameri243	20

Appendix A. Cross Section Availability and Mixture Definitions

coglex 6/9/87 716 entries coglex Page 34

alphabetical listing

material	page	material	page	material	page	material	page
americiu	21	benzene	28	c2h2cl2	24	cf250	22
antimony	11	beo	25	c2h3cl	23	cf251	22
ar	5	beoxide	25	c2h3o	23	cf252	22
argon	5	berke249	22	c4h3cl	24	ch	23
arseni74	7	berkeliu	22	c5h8o2	24	ch2	23
arsenic	8	berylli7	2	c6h6	28	ch2o	24
as	8	berylliu	2	c8h8	24	chlorine	5
as74	7	bi	16	ca	5	chromium	6
as75	8	bi209	16	cadmium	10	cl	5
astatine	16	bismuth	16	calcium	5	cm	22
at	16	bk	22	calif249	22	cm242	21
au	15	bk249	22	calif250	22	cm243	21
au197	15	boron	2	calif251	22	cm244	21
b	2	boron10	2	calif252	22	cm245	21
b10	2	boron11	2	californ	22	cm246	21
b11	2	br	8	carbon	3	cm247	21
ba	12	bromine	8	carbon12	2	cm248	21
ba138	12	c	3	carbon13	3	cncrt1	26
bariu138	12	c*3	30	cd	10	cncrt2	26
barium	12	c*9	30	ce	12	cncrt3	27
barytes	27	c*e	30	cerium	12	cncrt4	27
be	2	c10h16o5	24	cesium	12	cncrt5	27
be-metal	2	c10h8o4	24	cf	22	cncrt6	27
be7	2	c12	2	cf2	24	co	6
be9	2	c13	3	cf249	22	co59	6

alphabetical listing

Appendix A. Cross Section Availability and Mixture Definitions

alphabetical listing

material	page	material	page	material	page	material	page
1x041	29	nd	12	os	15	po	16
1x07	29	ne	4	osmium	15	polonium	16
1x09	29	neodymiu	12	oxygen	3	polyethy	23
1x10	29	neon	4	oy	29	polyform	24
magnesiu	4	neptu235	19	p	4	polystyr	23
manganes	6	neptu236	19	p31	4	potassiu	5
mercury	16	neptu238	19	pa	18	pr	12
mfecncrt	27	neptuniu	19	pa233	17	praseody	12
mg	4	ni	7	palladiu	10	promethi	13
mgnetite	27	ni58	7	paryl-c	24	prota233	17
mn	6	nickel	7	paryl-n	24	protacti	18
mn55	6	nickel58	7	pb	16	pt	15
mo	9	niobium	9	pbx9404	30	pu	20
mock-he	30	nitrog14	3	pbx9409	30	pu237	19
mockhe	30	nitrog15	3	pd	10	pu238	19
molybden	9	nitrogen	3	phosphor	4	pu239	20
mylar	24	np	19	platinum	15	pu240	20
n	3	np235	19	pluto237	19	pu241	20
n14	3	np236	19	pluto238	19	pu242	20
n15	3	np237	19	pluto239	20	pu243	20
na	4	np238	19	pluto240	20	pva	23
na23	4	o	3	pluto241	20	pvc	23
nai	25	o16	3	pluto243	20	quartz	25
nb	9	oralloy	29	plutoniu	20	ra	17
nb93	9	orcncrt	26	pm	13	radium	17

alphabetical listing

material	page	material	page	material	page	material	page
radon	16	silicon	4	th	17	u236o2	28
rb	8	silve107	10	th231	17	u237	18
re	15	silve109	10	th232	17	u237o2	28
re185	15	silver	10	th233	17	u238	18
re187	15	sio2	25	thallium	16	u238o2	28
res*abs	30	sm	13	thori231	17	u239	18
rh	10	sn	11	thori233	17	u239o2	28
rheni185	15	sodium	4	thorium	17	u240	19
rheni187	15	sr	8	thulium	14	u240o2	29
rhenium	15	ss304	26	ti	6	uo2	28
rhodium	10	steel	26	tin	11	urani233	18
rn	16	strontiu	8	titanium	6	urani234	18
ru	9	sulfur	5	tl	16	urani235	18
rubidium	8	sulfur32	5	tm	14	urani236	18
rutheniu	9	t	1	tritium	1	urani237	18
s	5	ta	14	tuballoy	29	urani238	18
s32	5	ta181	14	tungsten	14	urani239	18
samarium	13	tantalum	14	u	19	urani240	19
saran	24	tb	13	u233	18	uranium	19
sb	11	tc	9	u233o2	28	v	6
sc	5	te	11	u234	18	v51	6
scandium	5	techneti	9	u234o2	28	vanadi51	6
se	8	teflon	24	u235	18	vanadium	6
selenium	8	telluriu	11	u235o2	28	w	14
si	4	terbium	13	u236	18	water	26

Appendix A. Cross Section Availability and Mixture Definitions

alphabetical listing

material	page	material	page	material	page	material	page
xe	11						
xe134	11						
xenon	11						
xenon134	11						
y	9						
y88	8						
y89	9						
yb	14						
ytterbiu	14						
yttriu88	8						
yttrium	9						
zinc	7						
zirconiu	9						
zn	7						
zr	9						
zrh2	28						

Appendix B. Summary of Commonly Used Code Inputs

The following is presented for those who would like a quick reference list of COG inputs. It contains only the commonly used input statements and does not include explanations. The full descriptions with all options are explained in the body of this report. Refer to the index to find the approprate page number.

```
THIS IS THE TITLE INPUT


BASIC  NEUTRON  PHOTON  FEET  KEV  MINUTE


SURFACE

    1 PLANE  [x',y',z']1  [x',y',z']2
             (TR x0 y0 z0) (x1 y1 z1)  (x2 y2 z2)

    2 PLANE  [x',y',z']1  [x',y',z']2  [x',y',z']3
             [x',y',z']4  (TR....)

    3 BOX  ax  ay  az  (TR....)

    4 HEXAHEDRON  [x',y',z']1 .... [x',y',z']8
                  (TR....)

    5 PRISM  number-pts  [y',z']1  [y',z']2 ...
             [y',z']N {x'b1 x'b2}  (TR....)

    6 SPHERE  radius  (TR....)

    7 CYLINDER  radius  {x'b1 x'b2}  (TR....)

    8 CONE  a  radius  {x'b1 x'b2}  (TR....)

    9 TORUS  r  a  b  (TR....)

   10 REVOLUTION  number-pts  [x',r']1  [x',r']2
             ... [x',r']N  (TR....)
```

Appendix B. Summary of Commonly Used Code Inputs

```
GEOMETRY

   SECTOR 1 name-1 ±s1 ±s2 ±s3 OR ±s4 ±s5

   SECTOR 2 name-2 ±s6 ±s7

   UNIT   3 name-3 ±s8 ±s9 (TR....)

   BOUNDARY  VACUUM  ±s10 ±s11

   BOUNDARY  REFLECTION  ±s12 ±s13

   FILL  4

   DEFINE UNIT 3

      SECTOR 5 name-5 ±s14 ±s15

      SECTOR 6 name-6 ±s16 ±s17

      FILL  7


   PICTURE CS SECTOR xtl ytl ztl  xbl ybl zbl
                     xbr ybr zbr

   PICTURE P MATERIAL xc yc zc  r  d  θ  ϕ
                      m1 m2 m3 ...

   VOLUME REGION x0 y0 z0  x1 y1 z1  x2 y2 z2
                 length-x'  length-y'  length-z'


ASSIGN      s1 m1 r1 df1  (s2 m2 r2 dr2) ...

ASSIGN-M    s1 m1  (s2 m2) ...

ASSIGN-R    s1 r1  (s2 r2) ...

ASSIGN-D    s1 df1  (s2 df2) ...

ASSIGN-MD   s1 m1 df1  (s2 m2 dr2) ...

ASSIGN-ML   ma s1a s2a s3a ... (/ mb s1b s2b ...) ...

ASSIGN-RL   ra s1a s2a s3a ... (/ rb s1b s2b ...) ...
```

```
MIX  MAT=1 comp-1 spec-gravity-1
          (comp-2 spec-gravity-2) ...

     MAT=2 comp-1 spec-gravity-1 ...


SOURCE

   NPARTICLES=number

   DEFINE P=1  POINT  x0 y0 z0

   DEFINE P=2  LINE   x1 y1 z1  x2 y2 z2

   DEFINE P=3  SS-DISK  x1 y1 z1  x2 y2 z2
                r0 (rin)

   DEFINE P=4  CYLINDER  x1 y1 z1  x2 y2 z2
                r0 (rin)  {MATERIAL=number}

   DEFINE P=5  SPHERE  x1 y1 z1  r0 (rin)

   DEFINE E=1  NEUTRON LINE e1 intensity1
                (e2 intensity2) ...

   DEFINE E=2  PHOTON  DISTRIBUTION  e1 s1 e2 s2 ... sN
                {IMPORTANCE ....}

   DEFINE E=3  PHOTON  BIN  e1 s1 e2 s2 ... eN+1
                {IMPORTANCE ....}


   DEFINE T=1  DELTA  t0

   DEFINE T=2  GAUSSIAN  t0   δt  {IMPORTANCE ....}

   DEFINE T=3  DISTRIBUTION  t1 s1 t2 s2 ... sN
                {IMPORTANCE ....}

   DEFINE T=4  BIN  t1 s1 t2 s2 ... sN  {IMP..}

   DEFINE T=5  STEADY


   DEFINE A=1  uref vref wref  FIXED
```

Appendix B. Summary of Commonly Used Code Inputs

```
DEFINE A=2  POINT x y z ISOTROPIC
              {IMPORTANCE  μ1 i ...}

DEFINE A=3  NORMAL DISTRIBUTION  μ1 s1  μ2 s2 ... sN
              {IMPORTANCE ....}

DEFINE A=4  uref vref wref BIN  μ1 s1 ... μN+1 {IMP..}


INCREMENT 1.0E+10  P=1  E=1  T=1  A=1  I=0.7
INCREMENT 2.7E+10  P=2  E=3  T=1  A=4  I=1.5


WALK-AGE  NEUTRON REGION 1 TO 7 ENERGY 0.0 TO 0.5  tco
                                ENERGY 0.5 TO 1.0  tco
                  REGION 8      ENERGY 0.0 TO 1.0  tco
          PHOTON  REGION ALL    ENERGY 0.0 TO 20.  tco


WALK-PRODUCTION   ......  no production if particle,
                          region, energy included


WALK-SURVIVAL     ......  survival if particle,
                          region, energy included


WALK-FC           ......  forced collision if
                          particle,region, energy
                          included


WALK-COLLISION    ......  β = number out / number in


WALK-BC           ......  i (β = importance of region
                          being entered / importance
                          of region being left)


WALK-WT           ......  wtmin  wtmax
```

```
WALK-PS          ......   q  POINT  x y z
                          q  SPHERE  r  x y z
                          q  CYLINDER  r  x1 y1 z1 x2
                                          y2 z2
                          q  RING  r  x0 y0 z0 x1 y1 z1
                          q  DIRECTION  u v w

WALK-SDB         ......   b  POINT  x y z
                             and other options as
                             above

WALK-SEB         ......   e1 i1 ..... iN

ANALYSIS  NEUTRON  xt1 yt1 zt1  xbl ybl zbl  xbr ybr zbr
   thickness  (ENERGY=elower eupper)  (AGE=tlower tupper)
   (WEIGHT=wtlower wtupper)  (TITLE="....")

DETECTOR
   NUMBER 283  "Title...."
   REACTION  region  volume
   BOUNDARY  region 1  region2  area
   POINT  x  y  z  {LIMIT  REGION  number n
                   (REGION  number n) ... }
   DRF-E  NEUTRON  DOSE
   DRF-E  PHOTON  ENERGY-FLUX
   DRF-E  PHOTON  ENERGY-DEPOSITION  material
   DRF-E  NEUTRON  e1  drf1  e2  drf2  .... drfN
```

Appendix B. Summary of Commonly Used Code Inputs

```
BIN   T= t1   t2   t3 ....

BIN   E= NEUTRON   e1   e2   e3 ....

      A= uref   vref   wref     μ1     μ2    μ3 ....


END
```

Index

Index

Index

Index

Index

Revision History

Revision	Date	Description of changes
—	2Feb89	New manual. Replaces draft document dated October 10, 1986.