MITNE - 40

PULSE

An IBM 7094 Program for Calculation of Fast Neutron Kinetics by MonteCarlo

Progress Report, October, 1963

A.E. Profio

Massachusetts Institute of Technology
Department of Nuclear Engineering
Cambridge, Massachusetts

MITNE-40

PULSE

An IBM 7094 Program for Calculation of Fast Neutron Kinetics by MonteCarlo

Progress Report, October, 1963

A. E. Profio

Massachusetts Institute of Technology
Department of Nuclear Engineering

PULSE

Introduction

The program described in this report was written as part of the pulsed neutron source research program supported by National Science Foundation grants G-25049 and GP-1657. It has been tested and used on the IBM 7090 or 7094 computers at the Computation Center, M.I.T. However the code is in FORTRAN II and can easily be used at other installations. The program consists of a number of subroutines linked by a main program. The details of the coding are discussed below.

The object of the program is to calculate the density of neutrons as a function of position, velocity (or energy), and time in a single homogeneous finite medium of simple shape, for a time-dependent (pulsed) source. The history of each source neutron is followed by the usual Monte Carlo technique, using probability functions to describe what occurs at each collision, until the neutron either leaks out of the medium or is absorbed. There are no weighting or other biasing schemes employed. Neutrons making certain kinds of collisions, and lying within certain position, energy, and time intervals relative to the initial source burst time, are summed and stored. At the end of the computation these quantities are printed and punched on cards for subsequent analysis. There are, of course, statistical fluctuations in the values, and this is indeed the major limitation of the Monte Carlo calculation. A large number (10,000 or more) of histories must be followed for reliable results. The use of straight analog Monte Carlo is feasible because the program is designed for small or highly absorbing systems excited by fast neutrons, where the neutron makes only a few collisions on the average. The time for computation is essentially proportional to the mean lifetime in the medium, and computation of long lifetime systems is restricted by the time made available by the Computation Center.

Experimental work is proceeding on small (in terms of mean free paths) assemblies at the M.I.T. Rockefeller Generator. Lead has been investigated extensively, and later uranium blocks will be pulsed. The time-dependent leakage flux and time-dependent spectrum are measured. It is intended to compare the Monte Carlo calculations with the lead experiments, and then to use Monte Carlo to interpret the uranium experiments.

At the present time the program has been tested and results are available for a few lead assemblies. Recommendations for future work are given in the report.

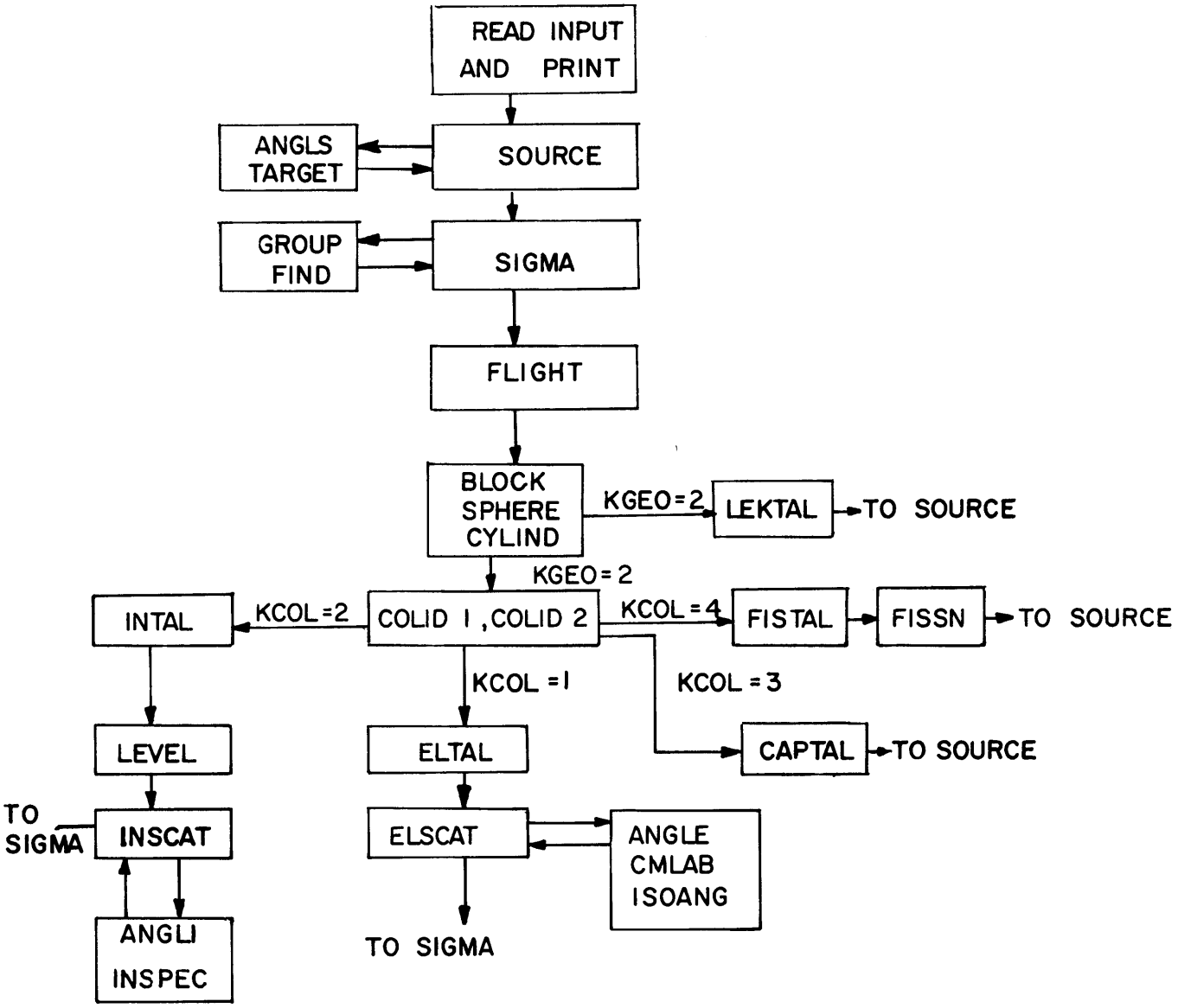
Overall Description

The operation of the program can be seen from the simplified flow diagram and the FORTRAN listings attached. The main program first reads each data card and prints its information for future reference. The information required on the data cards is discussed later, but includes the atomic density (either 1 or 2 constituents may be included) and pertinent microscopic cross sections. The main program first multiplies each cross section by the atomic density before storing the result and printing it. All other data is stored and printed as punched on the cards.

The main program starts the computation of each neutron history, and continues until it reaches the total number of histories specified in the data input. It then examines the results to see if any fissions have occured. If so, fission neutrons with some parameters (e.g. position where emitted) already computed are started out as for the original source neutrons, and the computation continues until the first cycle fission neutrons are exhausted. Some of these neutrons may cause further fissions and hence generate more neutrons, so that the computation proceeds until all fission neutrons are accounted for.

Considering only original source neutrons now, the program calls for the SOURCE subroutine. The input provides certain needed information including the type of source (discussed under the subroutine writeup). The subroutine returns the starting position

PULSE
FLOW DIAGRAM



(x, y, z coordinates), velocity, time (usually zero), and direction cosines (α, β, γ) of the source neutron. SOURCE may call on subroutine ANGLS in order to compute the angle with respect to the beam axis for an accelerator pulsed source outside of the medium. It may also call on subroutine TARGET, which is now a dummy subroutine but can be used to introduce a source distribution not covered by the options in SOURCE.

The program next uses subroutine SIGMA to calculate the cross sections for elastic scattering, inelastic scattering, capture or fission, and also the total mean free path and the probabilities for elastic or inelastic scattering, capture, and fission. An auxiliary subroutine, GROUP, is used to decide in which group the velocity lies; the velocities at the lower bound of each group are specified in the data input. SIGMA uses another subroutine, FIND, to interpolate between the cross sections given at the group boundaries. Linear interpolation is used and therefore the groups should be chosen so that none of the cross sections vary too much from a linear relationship between boundaries.

Main now calls FLIGHT and provides this subroutine with the velocity and the total mean free path. FLIGHT uses a pseudorandom number to select the free path from the exponential distribution of free paths. The subroutine also calculates the time at which the next collision will occur.

Next the program calls for one of three subroutines, depending on the shape of the assembly as specified in the input. The shapes available are a rectangular parallelepiped (subroutine BLØCK), sphere (subroutine SPHERE), or right circular cylinder (subroutine CYLIND). One of these subroutines calculates the new position for the neutron, using the free path distance given by FLIGHT, the previous position, and the direction cosines. The subroutine also decides if the new position is inside or outside of the block, sphere, or cylinder. If it is outside a leakage tally subroutine, LEKTAL, is called and one neutron is added to the appropriate register in a two dimensional array whose coordinates are a time group and an energy group. The initial time and group width (time interval) are specified in the

input; 100 time groups are allowed. Likewise the minimum energy recorded and the width of the energy groups are specified. Ten energy groups are allowed. Neutrons which leak out before or after the time tally interval or have an energy outside the energy tally range are so identified and summed in other registers (there seems to be some difficulty yet in this part of the program as the number tallied plus the number lying outside the time or energy ranges of interest do not add up to the number of source neutrons). The program now goes back to SOURCE and starts a new neutron history.

If the new position is still inside the assembly, subroutine COLID1 (if there is only one species of nucleus in the medium) or COLID2 (for two species) is called. A pseudorandom number is used to select one type of collision using the probabilities previously computed. If the collision is a capture, a tally subroutine named CAPTAL is called, and one neutron is tallied in the capture storage according to its time group. If the neutron lies outside the time range of interest this fact is also recorded.

If a fission occurs, subroutine FISTAL is used to tally in the time group, or record a neutron outside of the time range, as before. Another subroutine, FISSN, computes the number and velocities of the fission neutrons generated, and writes the parameters x, y, z, v, t for each neutron on tape for future use as explained above. When the fission neutron histories are started, an isotropic distribution of emission of fission neutrons is generated in SOURCE. Following the tape operations the control is returned to main which starts a new neutron.

If COLID1 (or COLID2) selected an inelastic scattering collision, subroutine INTAL is called. This subroutine again tabulates a neutron in a time group or notes if it is outside the range. The subroutine LEVEL first calculates the cross section for the particular velocity neutron scattering from each of the inelastic scattering levels specified in the input (up to 20 allowed), interpolating between the cross sections given at each of the velocity group boundaries. Then the probabilities for scattering by each level are computed. These probabilities are supplied to subroutine INSCAT.

According to a number supplied by the input, INSCAT calculates either an isotropic angular distribution or an anisotropic one, with the help of subroutine ANGLI. At the present time ANGLI puts in straight ahead scattering in the center of mass system, but the program could be rewritten for other distributions. Returning to INSCAT, if the velocity of the incident neutron is below a certain velocity specified in the input, scattering by individual levels is assumed. A pseudorandom number is used to select the level according to the probabilities previously calculated. The velocity of the inelastically scattered neutron is found by subtracting the velocity of the level from the incident velocity (the level velocities are also specified in the input). If the incident velocity is above the specified cutoff, the scattered velocity is selected from a Maxwell-Boltzmann distribution using a 22 point cumulative probability table included with the data cards. The probabilities are normalized and the most probable velocity of the distribution is calculated from a given constant times the incident velocity. The subroutine which does this is named INSPEC. Main now returns to SIGMA since the parameters x, y, z, v, t, α, β, γ of the inelastically scattered neutron have been computed, and the history of the scattered neutron is continued.

If COLID1 or COLID2 selects an elastic scattering collision, subroutine ELTAL is called. In this case the collision is tallied in a time group and also in a space group (one of ten distance intervals inside the assembly along the z-coordinate). This part of the prgram should be checked further since there is some indication that the assignment to the proper space group may be somewhat imperfect. Again if the collision occured outside the time range of interest, the fact is noted. The subroutine also checks that the collision occured inside the assembly, although if the program runs correctly this could not occur since the history would have terminated in LEKTAL. In any event, the prgram proceeds to subroutine ELSCAT, where the velocity is first compared with a certain velocity specified in the input. If the velocity is less than the cutoff (which is a function of the particular nuclide scattering), isotropic elastic scattering is assumed. An isotropic angular distribution is generated with the help of pseudorandom numbers, using

also the subroutine ISOANG. If the incident velocity is above the cutoff, an anisotropic direction cosine γ in the center of mass system is selected from a cumulative probability table given in the input, using subroutine ANGLE. The program provides for a ten point specification of the probability function, for each of the twenty velocity groups used in this Monte Carlo program. Thus the elastic scattering angular distribution can vary with the velocity of the incident neutron. ELSCAT also calculates the velocity of the scattered neutron. If the mass number of the scattering nucleus is above a certain number given in the input, the velocity is set equal to the incident velocity. If on the other hand the mass number is below the cutoff, the velocity (in the lab system) will be lower. Subroutine CMLAB calculates the direction cosines in the lab system and the velocity of the scattering nucleus. From ELSCAT the program transfers to SIGMA and the history of the neutron is continued.

The main program performs a few other functions which will now be discussed. The program listing can be followed for details. It rewinds the tapes on which fission neutrons may be recorded, and sets various registers to zero. It checks to see if the source neutron velocity lies within the group structure set up in the input (group index 1 through 20). SIGMA returns an index equal to zero if the neutron lies below the lowest group boundary, and since no cross sections are available for such neutrons, the source velocity cannot be this low. Five trials are made and if an acceptable source neutron is not found the program calls EXIT and terminates with a printout of the number of trials, NT=5. Main also checks that the free path to the next collision, DIST, is a zero or positive number. If it is not then the program exits. The same check is made on the time variable TIME. The main program keeps track of the number of times the same neutron makes a scattering collision (either elastic or inelastic), the integer variable named KSCAT. If more than 100 scatterings are made, the neutron is discarded (but the number of such neutrons is counted in NOSL) and a new neutron is started. This prevents following an unusual neutron for very long times.

The fission neutron cycling described previously is accomplished by first setting a variable named MULT to 1. At the end of each neutron history the main program examines MULT. If MULT equals 1, the program continues with the source neutrons. When all of these have been exhausted, the SOURCE option is set to the internal isotropic source option, MULT is set to 2, the fission tapes are rewound, the neutron parameters are read from the first tape (designated by a number for variable KT1), computations proceed, and any new fission neutrons are written on the second tape (whose number is specified by the value assigned to KT2). When the program examines MULT it follows this tape reading and writing sequence. Finally when all neutrons on tape KT1 are exhausted, MULT is set to 3, the tapes are rewound, and the second tape is read while over writing now on the first tape. This procedure continues until all neutrons are exhaused, and the output portion of the main program is reached. It should be noted that this whole sequence of using tapes has not been tested as yet. Output is discussed later.

Subroutines

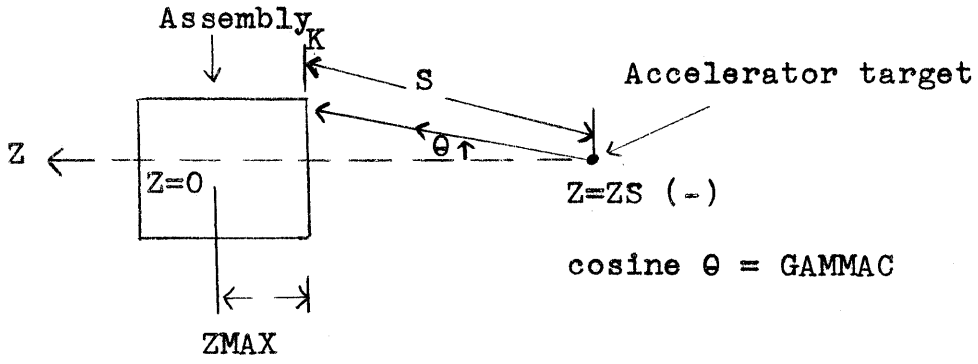
SOURCE

Four source options are built into this subroutine. If an input variable KS is set equal to 1, the position X, Y, Z is set equal to position XS, YS, ZS provided by the program calling SOURCE (these are read from input cards for the original source neutrons, or from the fission neutron tape). The source angular distribution is isotropic. The velocity is as specified by a variable named PARA in the input or fission tape, and time by a variable THETA.

If KS=2, the second option is chosen. This generates X and Y uniformly over a range $\pm$ XS and $\pm$ YS, while Z is set to ZS. Thus if XS and YS are set to the half-thicknesses XMAX and YMAX (or radius of a sphere or cylinder), this option can be used to generate a uniform plane source on one face or even inside a block (or on a plane near a sphere or cylinder). The direction cosines ALPHA and BETA are set to zero and the direction cosine GAMMA is set to 1.0,

so this represents a plane monodirectional source incident. The velocity VEL is given by a fixed input value PARA less a input constant PARB times a pseudorandom number between 0 and 1 (generated by a subroutine RANNØF in the library of the M.I.T. Computation Center computer). Thus option 2 provides for a source velocity spread due, e.g., to energy loss of ions in the pulsed accelerator target. TIME is set equal to zero (impulse source).

Option 3, with KS=3, computes the velocity from the same expression as for option 2, with an additional negative term. This term is given by an input constant PARC time (1 - GAMMAC) where GAMMAC is the direction cosine (0 to 1) which the neutron makes with the beam axis, and is obtained from the subroutine ANGLS described below. A neutron emitted at an angle with the beam (+z direction) has a lower velocity, which can be approximated by a linear function. The other direction cosines ALPHA and BETA are computed by ISOANG. The source geometry which this option is intended to represent is shown in the sketch.



The position at which the neutron enters the assembly is given by X = S times ALPHA, Y = S times BETA, and Z = -ZMAX. The time varies with the free flight distance S and the velocity, and is computed for each neutron. Zero time is taken as the instant of emission from the target (impulse source).

Option 4, KS=4, is simply to call the subroutine TARGET so that other source configurations can be included if desired. SOURCE has been debugged on several cases for KS=1, 2, and 3.

TARGET

So far this is essentially a dummy subroutine which returns the position and direction cosines specified in its call statement. It does set TIME = 0.0 and calculates a velocity like option 2 except an additional negative term is included. This term consists of a constant PARC times the absolute value of GAMMA given in the call statement. TARGET was not tested completely but is believed to be all right.

ANGLS

An anisotropic distribution for GAMMAC is computed, using a pseudorandom number to select a value for GAMMAC from a cumulative probability distribution given by the ten point array named SP. ANGLS was debugged on a known distribution.

SIGMA

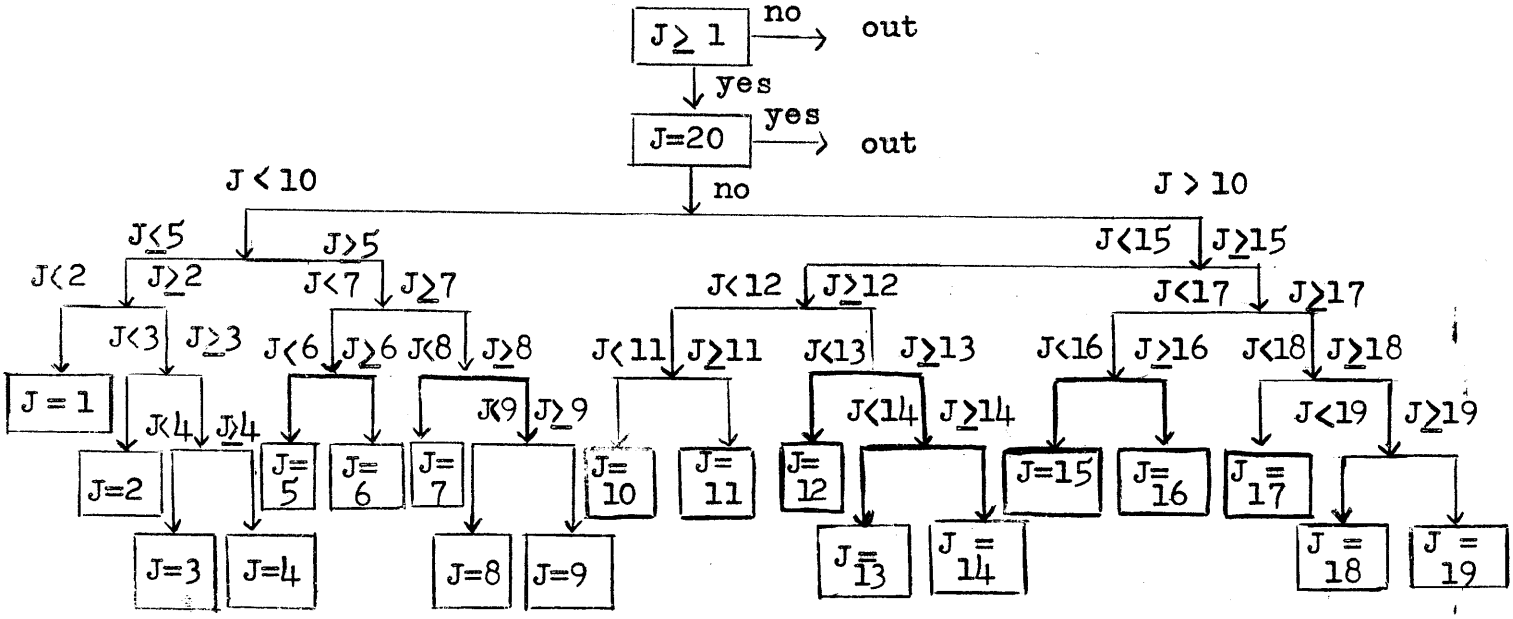
The input gives values for the cross sections at the lower limit of the group boundaries, the variable cross section names are of the form SBX 1 or SBX 2 (referring to nuclide 1 or 2). The group boundary variable is named EBOUND but it is actually the velocity which is punched on the card. The variable names are really dummies in FORTRAN and this change does not matter. If the atomic density AD2=0.0 in the input, only one nuclide is considered in the program. Otherwise the cross sections SX1, SX2 are calculated each time for each nuclide, for velocity VEL (again written as EN in this subroutine in FORTRAN, but the name is without significance). SE indicates elastic, SI inelastic, SC capture, SF fission, etc. The total mean free path TMFP is computed from the reciprocal of the total macroscopic cross section. The probabilities PX1, PX2 for elastic, inelastic, capture or fission interactions for each nuclide (1 or 2) are computed by TMFP times the partial cross section. This is equivalent to, e.g. calculating the probability of elastic scattering in nuclide 1 by $\Sigma_{(elastic,1)}/\Sigma_{total}$. The elastic scattering, inelastic scattering, and fission probabilities are calculated directly, and the capture probability is calculated by the difference between unity and the sum of these probabilities. If the capture probability turns out to be less than 0.0001, it is set equal to zero. If two nuclides are involved,

all probabilities except the capture probability for nuclide 2 is assumed if no other process occurs according to the probability selection by the pseudorandom number. SIGMA was debugged by running 50 neutrons of different velocity and seeing that the proper numbers were calculated.

FIND is an interpolation function and was debugged with SIGMA.

GROUP

The velocity (named EN here) and the velocity of the lower limit of each group (named EBOUND here) are given as input. EBOUND is an array with index J, the group number. The velocity is within group J if it is equal to larger than EBOUND (J) but not equal to or larger than EBOUND (J+1). If the velocity is less than EBOUND (1) then it is not in any group and a code number KGP=1 is returned. If this does not occur, the subroutine proceeds to see if the velocity is larger than EBOUND (20). If it is, it is in group 20 and the control returns to main with a code KGP=2 indicating that the neutron is in a group, and the integer variable J=20 is also returned. If the velocity lies between EBOUND (1) and EBOUND (20), a binary search is initiated. The search pattern is shown in the diagram below. In the end the proper group index J is found and returned, together with KGP=2. In EN≡EBØUND (J) then J is found immediately, but this is not shown in the diagram.



GROUP was debugged by putting in 75 velocities and seeing that they were sorted into the proper groups.

FLIGHT

A pseudorandom number is used to select from an exponential distribution the number of mean free paths which a neutron will travel before its next collision. The pseudorandom number is first examined to be sure it is positive and larger than 0.0000454. This corresponds to rejecting any free paths larger than 10 mean free paths. If this occurs, a new pseudorandom number is generated and the computation proceeds. In addition to the free flight distance d (named DIST) the subroutine updates the time by $t' = t + d/v$. FLIGHT was debugged by checking the distribution of DIST and TIME with 100 starting neutrons.

BLOCK

The new position is calculated from the old coordinates, the free flight distance, and the direction cosines. The input includes the half-thickness in each coordinate, XMAX, YMAX, ZMAX. To see if the new position is outside the block, the absolute value of each coordinate is compared with the corresponding half-thickness. For example, of $|X| > XMAX$ the neutron is outside; if not Y and then Z are compared to YMAX and ZMAX respectively. The new coordinates are returned to the main program along with a code number KGEO=1 if the position is inside the block, or KGEO=2 if the position is outside. BLOCK was tested with ISOANG and 100 neutrons to see that the inside-outside decision was made properly.

SPHERE

The new position in x, y, z coordinates is calculated as for BLOCK. The radius of the sphere, RMAX, is given in the input. If $X^2 + Y^2 + Z^2$ is less than $(RMAX)^2$ the new position is inside the sphere, and if not it is outside. X,Y,Z are returned as well as KGEO as for BLOCK. SPHERE was debugged by running 100 neutrons half inside and half outside the sphere, and checking that the subroutine gave the proper propertion.

CYLIND

The new position in x, y, z coordinates is calculated as for BLOCK. The Z coordinate is taken along the axis of the cylinder. The half-height ZMAX and the radius RMAX are specified in the input. The subroutine decides if $|Z| < ZMAX$ and if $X^2 + Y^2 < (RMAX)^2$. If so, the new position is inside and KGEO=1, otherwise it is outside and KGED=2. CYLIND was debugged as for SPHERE.

COLID 1

First a pseudorandom number R is generated with RANNOF, as usual. Then the previously computed probability for elastic scattering, PE1, is subtracted from R. If the difference is negative than an elastic collision has occured, and a code variable KCOL=11 is returned. If not, R-PE1-PI1 is calculated (PI1 is the inelastic scattering probability) and again if a negative result is obtained the collision is inelastic and KCOL is set to 21. If not, the fission probability PF1 is tested and if a fission collision has occured, KCOL=31. If none of these tests give a negative result then capture is assumed and KCOL=41. It can be shown that this procedure is equivalent to selecting the type of collision, where PE1 + PI1 + PF2 + PC1 = 1, according to the probability for each type. The main program later uses KCOL to extract the nuclide number KNUCL=1 and the collision type, KTYPE = 1, 2, 3 or 4. COLID1 was debugged by running 1000 cases with given probabilities and seeing that the distribution calculated was the same within statistics.

COLID 2

In case two nuclides (or two sets of cross sections and atomic densities) are present, COLID 2 is called. The computation for the nuclide and the type of collision proceeds as for COLID 1, where the probabilities PE1, PE2, PI1, PI2, PF1, PF2, and PC1 are successively subtracted starting with a pseudorandom number between 0 and 1. If none of these computations meet the test, capture in nuclide 2 is assumed. The order of subtraction is chosen as given because elastic scattering is usually the most common event, then inelastic or fission, and capture is often small. The code variable KCOL is a two digit number with the first digit 1, 2, 3 or 4 according to whether elastic, inelastic, fission, or capture was selected. The second digit is 1 or

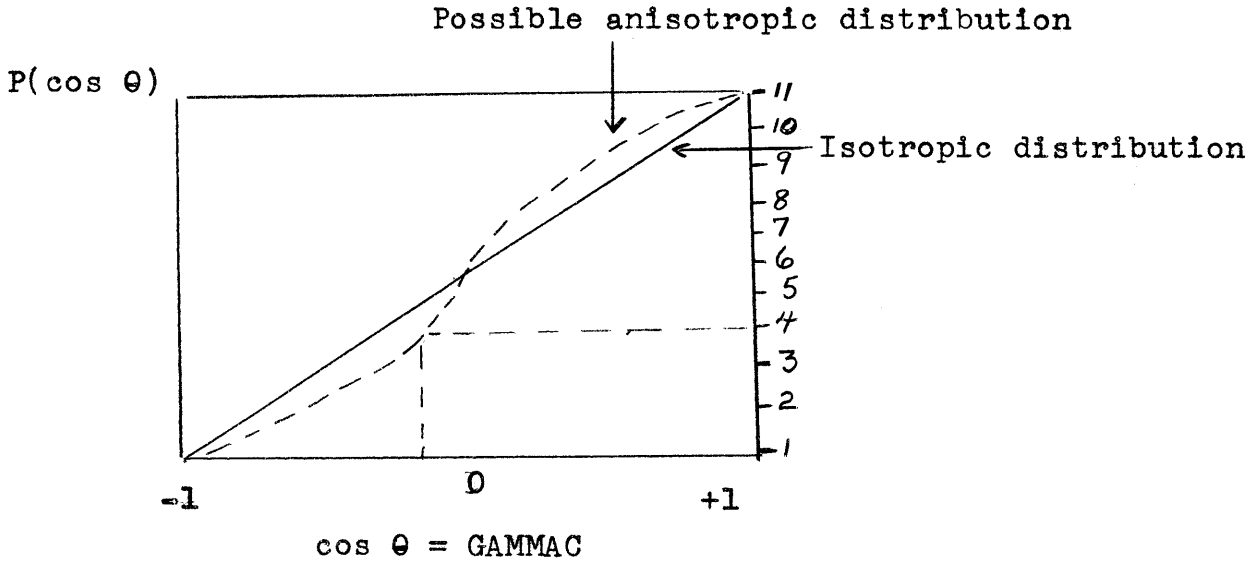
2 according to whether the collision was with nuclide 1 or 2. COLID 2 was debugged as for COLID 1.

ELSCAT

The velocity VEL is compared with the input variable SLIM which gives the upper velocity limit for which isotropic center of mass system scattering can be assumed. If VEL is equal to or greater than SLIM, anisotropic scattering occurs and the subroutine calls ANGLE. If isotropic scattering occurs, the center of mass direction cosine GAMMAC is computed by GAMMAC=2R-1 where R is a pseudorandom number between 0 and 1. The computation gives a pseudorandom number between -1 and +1. Next the subroutine decides if the mass number of the scatterer, A is equal to or greater than a limiting number ALIM specified in the input. If it is, then the nucleus is assumed to be so heavy that negligible moderation occurs and the lab system is equivalent to the center of mass system. ISOANG is called and supplied with GAMMAC from either ANGLE or the random number between -1 to +1. ISOANG generates α, β, γ as described later. If $A < ALIM$, a transformation from center of mass to lab coordinates is performed by CMLAB.

ANGLE

A pseudorandom number between 0 and 1 is generated, and is then converted to a integer between 1 and 11 and a remainder. These are then used to interpolate a value for GAMMAC between -1 and +1, from a cumulative probability table. The cumulative probability at GAMMAC=w is the probability that GAMMAC lies between GAMMAC=$w_{minimum}$ (-1 in our case) and GAMMAC=w. The probability is 100% that it lies between $w_{minimum}$ and $w_{maximum}$(+1 in our case). Thus the differential elastic scattering cross section expressed as $\sigma(\cos\Theta) = \sigma(GAMMAC)$ is integrated from $\cos\Theta = -1$ to $\cos\Theta$, and normalized by dividing by $\sigma(\cos\Theta = +1)$. This is now the integral probability $P(\cos\Theta)$ which is plotted against $\cos\Theta$ as shown in the figure below.



In the program, the ordinate P is divided into 10 equal intervals. The points are labeled from 1 to 11 because of the way arrays are indexed in FORTRAN. The value for GAMMAC at 11 is always taken as + 1.0 so it need not be given in the data. The value of GAMMAC at the other ten points (1 through 10) are given in an input array named AP. When the program is run, a value for GAMMAC is linearly interpolated between the values given in the array, using the random number to select an ordinate, as mentioned above. ANGLE was debugged by putting in two different angular distributions, running 500 or 100 histories, and noting that the distributions computed by Monte Carlo fit the assumed distributions within statistics. However further tests would be in order.

ISOANG

The polar direction cosine GAMMAC is supplied to this subroutine. Two pseudorandom numbers R1 and R2, between 0 and 1, are generated. Then the direction cosine GAMMA=GAMMAC, but ALPHA and BETA are to be chosen so that the azimuthal angular distribution is isotropic and so $\alpha^2+\beta^2+\gamma^2 = 1$. The subroutine does this by solving the following equations

$$\alpha = \xi_1 \sqrt{\frac{1-\gamma^2}{\eta}}$$

$$\beta = \xi_2 \sqrt{\frac{1 - \gamma^2}{\eta}}$$

where $\xi_1 = 2R_1 - 1$ and is a pseudorandom number between 0 and 1,

$\xi_2 = 2R_2 - 1$

$\eta = \xi_1^2 + \xi_2^2$

The subroutine sets VEL equal to the velocity given in the call statement. ISOANG was tested with BLOCK on 100 neutrons and found to give the proper distribution as nearly as could be decided by examination.

CMLAB

The conversion from center of mass to laboratory coordinates is made by solving the following equations. The subscript c indicates center of mass system, p an intermediate coordinate system, and no subscript refers to the lab system. The equations are derived, e.g. in the Clark and Hansen notes (see general references). As before ξ_1, ξ_2 are pseudorandom numbers between -1 and +1, and $\eta = \xi_1^2 + \xi_2^2$. GAMMAC=γ_c is given in the call statement. α_i, β_i γ_i are the input (lab system) direction cosines also given in the call statement.

$$\alpha_c = \xi_1 \sqrt{\frac{1 - \gamma_c^2}{\eta}} \qquad \beta_c = \xi_2 \sqrt{\frac{1 - \gamma_c^2}{\eta}} \qquad \gamma_c = \gamma_c$$

$$\alpha_p = \frac{\alpha_i \gamma_i \alpha_c - \beta_i \beta_c}{\sqrt{1 - \gamma_i^2}} + \alpha_i \gamma_c$$

$$\beta_p = \frac{\beta_i \gamma_i \alpha_c + \alpha_i \beta_c}{\sqrt{1 - \gamma_i^2}} + \beta_i \gamma_c$$

$$\gamma_p = -\alpha_c \sqrt{1 - \gamma_i^2} + \gamma_i \gamma_c$$

The desired direction cosines in the lab system after the collision are then

$$\alpha = \frac{\alpha_i + A\alpha_p}{\sqrt{1+A^2+2A\gamma_c}} \qquad \beta = \frac{\beta_i + A\beta_p}{\sqrt{1+A^2+2A\gamma_c}}$$

$$\gamma = \frac{\gamma_i + A\gamma_p}{\sqrt{1+A^2+2A\gamma_c}}$$

where A is the mass number, and the lab system velocity after the collision is

$$v = \frac{v_i \sqrt{1+A^2+2A\gamma_c}}{A+1}$$

CMLAB is believed to be debugged but was not tested thoroughly because of a small error in the testing program. The subroutine should be tested further.

LEVEL

This subroutine first linearly interpolates between the cross section SBL(L) for inelastic scattering from different levels (given at the 20 velocity group boundaries) to give SL(L). The latter is the cross section for the level scattering at the particular incident velocity, with L being the index (1 to 20) for the level. Levels are numbered from the lowest to the highest. In case the velocity is in group 20 no interpolation is required since all cross sections are assumed constant above VBOUND(20), the velocity at the lower limit of group 20. The inelastic scattering cross sections do not have to be normalized in the data input, because LEVEL computes scattering probabilities PL(L) for each level from $PL(L) = \frac{SL(L)}{\sum_{L=1}^{20} SL(L)}$.

LEVEL has been debugged by putting in two different velocities and checking that the proper PL's were calculated.

INSCAT

An input constant KIA=1 signifies that inelastic scattering is assumed isotropic, while KIA=2 signifies the scattering is anisotropic. If isotropic, $\gamma_c = \xi$, a pseudorandom number between -1 and +1, and α and β are computed by ISOANG. If anisotropic, ANGLI is called to get γ_c and then ISOANG to get α and β. Next if the velocity is less than the velocity specified for VST, individual level scattering occurs and the level which does the scattering is computed with a pseudorandom number and the probabilities PL(L). The new velocity

is the incident velocity less the velocity of the level, VL(L) specified in the input. If the incident velocity was equal to or greater than VST, the subroutine INSPEC is called to obtain the scattered velocity. INSCAT was debugged on test cases, but then recompiled to include some corrections for use with INSPEC. It should be tested further.

ANGLI

At the present time ANGLI sets GAMMAC=1.0, but is called with the incident velocity and the mass number A so that it could be rewritten to include an angular distribution for inelastic scattering which depends on these parameters. ANGLI has not been used.

INSPEC

The most probable velocity VMAX of a Maxwellian distribution for inelastically scattered neutrons is found from an input constant CIN times the incident velocity. A pseudorandom number is used to select a value for v'/v_{max} from a normalized (and slightly modified) Maxwell-Boltzmann velocity distribution, where v' is the scattered velocity. A 22 point cumulative probability table for v/v_{max} is included in the data and is given as an array P(K). The values used are given below

K	P(K) = v/v_{max}	K	P(K) = v/v_{max}
1	0.20	12	1.16
2	0.43	13	1.22
3	0.54	14	1.30
4	0.64	15	1.37
5	0.72	16	1.45
6	0.78	17	1.53
7	0.85	18	1.66
8	0.91	19	1.80
9	0.97	20	2.04
10	1.04	21	3.00
11	1.10	22	3.00

Any other velocity distribution could be included by substituting a new P(K) array. INSPEC was debugged but should be tested further because of a minor error in the testing program.

FISSN

The (integral) number of fission neutrons emitted is found by first calculating the average $\bar{\nu}$ for the incident velocity,

$$\bar{\nu} = \nu_f + \delta \cdot v^2$$

The input contains the constant term ν_f = FNU and the coefficient of the (velocity)^2; δ = DELNU. These are available for the various fissionable nuclides. The range of $\bar{\nu}$ is between 2 and 4 from thermal to the highest velocities of interest to us. The subroutine decides if the calculated $\bar{\nu}$ is between 3 and 4 or between 2 and 3. If the former is true, the subroutine generates either 3 or 4 neutrons each time, with the help of a pseudorandom number so that on the average the number is $\bar{\nu}$. Likewise the subroutine generates either 2 or 3 neutrons each time so $\bar{\nu}$ is the average, if $2 < \bar{\nu} < 3$. Next a different velocity is selected for each of the fission neutrons, as selected from a cumulative probability table with the help of pseudorandom numbers. The cumulative probability table is set up in a way similar to that in ANGLE, but 21 points are used. The array giving the fission neutron velocity at each of the 21 points is named FP(K) with K=1 through 21. The velocity is linearly interpolated as usual. The subroutine sets the variable PARA to the chosen velocity, THETA=TIME, XS=X, YS=Y, ZS=Z and writes these on the tape KT1 or KT2 as specified in the value of KT given in the call statement. Finally the subroutine adds one count to the integer variable NF, which is the number of fission neutrons whose parameter are written on the tape. FISSN has been debugged and tested but should be further tested with main.

LEKTAL

This subroutine sets up a two dimensional array LEAK (ITIME, IEN) where ITIME is an integer between 1 and 100 and IEN is an integer between 1 and 10. The array is ordered such that the first index varies most rapidly and the second least rapidly. That is, all the values of LEAK for ITIME=1 to 100 are arranged in order going from left to right and top to bottom, for IEN=1. Then all the values LEAK(1→100, 2) are written, etc. LEAK is the number of neutrons leaking from the assembly, i.e. the number whose

position ends up outside the assembly when it is recomputed for the next collision. ITIME in an integer time variable,

$$ITIME = (TIME - TD)/TCH$$

where TIME is the time at which the next collision would have occured, TD is an input constant corresponding to a time delay from the zero (burst) time, and TCH is an input constant for the time channel, or time group, width. TD and TCH are given in nanoseconds (10^{-9}sec). IEN is an energy group number,

$$IEN = \frac{0.5227 \times (VEL)^2 - EMIN}{ECH}$$

The velocity is first converted to an energy. EMIN is an input constant giving the minimum energy to be tallied, in Mev. ECH is an input constant giving the energy group width, in Mev. If the computed ITIME < 1 then the time is less than the time delay; a code variable KLEK is set equal to 1 and the control returns to the main program. Likewise if ITIME is greater than 100, the time is greater than the time tallied and KLEK=2. If the energy is less than EMIN, KLEK=3 and if it is greater than the maximum energy tallied, KLEK=4. The main program later adds one neutron to one of the variables given below, as appropriate:

NLTD	no. less than time delay
NGTR	no. greater than time range
NLME	no. less than minimum energy
NGER	no. greater than energy range

If the neutron is tallied in LEAK then KLEK=5. LEKTAL has been debugged in test cases and in running with main.

ELTAL

Elastic scattering collisions are tallied in the array NELS (ITIME,IZ) where as before ITIME is the time group index from 1 to 100 and now IZ is a Z coordinate interval index from 1 to 10.

$$IZ = 6.0 + \frac{5.0(Z)}{ZMAX}$$

which is intended to give IZ=1 for a collision with $-0.8(ZMAX) > Z \geq -ZMAX$

(algebraically), IZ=6 for $0.2(ZMAX) > Z \geq 0$, IZ=10 for $0.8(ZMAX) \leq Z <$ ZMAX and similarly in between. If the time is less than TD, a code number KELS is set to 1 and later one is added to NLTD. If the time is greater than the time range, KELS=2 and one is later added to NGTR. If Z is somehow less than -ZMAX or greater than +ZMAX then KEL=3 or 4, respectively, and in either case one is later added to the variable NGZR. If the collision is within the ranges, the proper tally is made in NELS and KELS=5.

INTAL

Inelastic scattering collisions are tallied in NINS (ITIME). If the time of scattering is less than TD, a code number KINS=1. If greater than the time range, KINS=2. Later one is added to NLTD or NGTR, respectively. If the scattering is tallied, one is added to NINS and KINS=5. INTAL was debugged by putting in 1000 random numbers in TIME and seeing that the NINS distribution was uniform except for statistics.

CAPTAL

Captures occuring within the time range tallied are added to the array KAPT(ITIME), and the code KCAP=5. If the time is less than TD, KCAP=1 and one is later added to NLTD. If the capture occured outside the time range, KCAP=2 and one is later added to NGTR. CAPTAL was debugged by putting in 1000 random numbers for TIME and seeing that the KAPT distribution was uniform within statistics.

FISTAL

Fissions are tallied in NFIS(ITIME), KFIS=5 or are added later to NLTD or NGTR with KFIS=1 or KFIS=2, respectively.

Arrangement of Deck

I.D. card (as prescribed by the Computation Center)

* XEQ

BINARY

The binary deck

* DATA

The data cards

Data Card No.	Values for	Format
1	XS, YS, ZS, PARA, PARB, PARC, THETA, KS, NEUT Format statements are explained in the IBM and Computation Center memos. XS, YS, ZS are the source coordinates (cm), PARA, PARE, PARC allow specification of source velocity (units of 10^9 cm/sec), THETA is the source time in nanoseconds (usually 0.0000), KS specifies the source option, NEUT is an integer variable giving the number of neutron histories to be computed in this run.	(7F8.4,I2,I4)
2	SP (1-10) The array specifying an anisotropic source distribution. The values punched are values of $\gamma = \cos \Theta$ for an index M.	(10F7.4)
3	XMAX, YMAX, ZMAX, RMAX, KAS The dimensions of the assembly (cm) and the shape (KAS = 1 for block, 2 for cylinder, 3 for sphere).	(4F8.4, I2)
4	TD, TCH, EMIN, ECH, KT1, KT2 The time delay (nanoseconds), time channel width (nsec), minimum tallied energy (Mev), energy channel width (Mev), and logical tape numbers for the fission subroutine (allowable numbers are specified by the Computation Center).	(4F7.3,2I3)
5	P (1-11)	(11F6.2)
6	P (12-22) The array specifying a Maxwell-Boltzmann distribution for inelastic scattered velocity. The values punched are normalized velocities for an index K.	(11F6.2)
7	VBØUND (1-10)	(10F7.4)

Card No.	Values for	Format
8	VBØUND (11-20) The velocities at the lower limt of each velocity group for cross section calculations (in units of 10^9 cm/sec).	(10F7.4)
9	AD1, A1, ALIM1, SLIM1, CIN1, VST1, FNU1, DELNU1, KIA1 For nuclide 1, the atomic density (in units of 10^{24} cm^{-3}), the mass number, the mass number below which a center of mass to lab system conversion must be made, the velocity above which anisotropic center-of-mass elastic scattering must be assumed, a decimal number used in subroutine INSPEC, a decimal number specifying the velocity below which individual level inelastic scattering occurs, two decimal numbers used in subroutine FISSN, and KIA = 1 if isotropic c.m.s. inelastic scatter or = 2 if anisotropic.	(F7.5, 2F7.2, 5F8.4, I2)
10	SBE 1 (1-10)	(10F7.3)
11	SBE (11-20) For nuclide 1, cross sections (barns, 10^{-24} cm^2) for elastic scattering at the velocities VBØUND.	(10F7.3)
12,13	SBI 1 (inelastic scattering cross sections)	(10F7.3)
14,15	SBF 1 (fission cross sections)	(10F7.3)
16,17	SBC 1 (capture cross sections)	(10F7.3)
18	AP 1 (1-10,1)	(10F7.3)
19 through 37	AP 1 (1-10, 2-20) The array specifying the angular distribution in anisotropic elastic scattering for (M,J) where M	

Card No.	Values for	Format
	is an index and J is the velocity group number. The values punched are values of $\gamma = \cos \Theta$.	
38	VL 1 (1-10)	(10F7.3)
39	VL 1 (11-20) Velocities (in units of 10^9 cm/sec) for up to 20 inelastic scattering levels (index L, numbered from lowest to highest velocity)	(10F7.3)
40 through 79	SBL 1 (1-10,1) through SBL (11-20,20) For nuclide 1, cross sections (barns) for inelastic scattering for (L,J) where L is the level number (lowest to highest) and J the velocity group number.	(10F7.3)
80,81	FP 1 (1-11), FP 1 (12-22) The array for the fission neutron velocity (units of 10^9 cm/sec) versus index K, used in subroutine FISSN.	(11F6.3)
82	AD2, A2, ALIM2, SLIM2, CIN2, VST2, FNU2, DELNU2, KIA2 Same as card 9 except for nuclide 2. If AD2 = 0.0 the rest of the card need not be punched and this will be the last card in the deck.	(F7.5, 2F7.2, 5F8.4, I2)
83-	If AD2 $\neq$ 0.0, the material constants for nuclide 2, as for cards 10 through 81 for nuclide 1.	

Summary of Normal Output

The data card values, labeled by the appropriate variable name, except for the cross sections SBE, SBI, SBF, SBC which are the macroscopic cross sections.

NL, the number leaking out
NC, the number captured
NS, the number scattered (elastic or inelastic)
NF, the number of fission neutrons
NLTD, the number ending up at less than time TD
NGTR, the number ending up over the time range
NGZR, the number from ELTAL which are outside the Z range
NLME, the number ending up at less than EMIN
NGER, the number ending up outside the energy tally range
NOSL, the number scattering more than 100 times and so dropped
LEAK, the number leaking for ITIME 1-100, IEN=1 to ITIME 1-100, IEN=10
NELS, the number of elastic scatters for ITIME 1-100, IZ=1 to ITIME 1-100, IZ = 10
NINS, the number of inelastic scatters for ITIME 1-100
NFIS, the number of fissions for ITIME 1-100
KAPT, the number of captures for ITIME 1-100

Preliminary Results

Preliminary results are available for lead (atomic density 0.03290×10^{24} cm^{-3}, mass 207.0), at source velocity 1.265×10^{9} cm/sec, giving an elastic scattering macroscopic cross section of 0.185 cm^{-1}. The angular distribution for elastic scattering was taken from ANL-5567 or BNL-400 (2nd Ed). The derived distributions are slightly different, at least partly due to difficulty in extrapolation. However in a test case the time constant was indistinguishable. The complete data would have to be examined for the details, but the cases calculated so far can be summarized as follows (capture and fission and zero):

Dimensions (cm)	Neutrons followed	Σ_{in} cm^{-1}	τ (nsec)
20.32 x 20.32 x 20.32	10000	0.0	9.5
40.64 x 40.64 x 40.64	10000	0.0	28
10.16 x 71.12 x 81.28	30000	0.002	9.3
20.32 x 71.12 x 81.28	10000	0.002	23

General References

M.I.T. Computation Center Handbook and Memos

IBM Bulletins on FORTRAN and the IBM 7090

M. Clark and K. F. Hansen, 22.53 Digital Computers in Nuclear Engineering, (Class Notes, M.I.T.)

J. J. Loechler and J. E. MacDonald, AEC Report APEX-706, 1961

H. Rief, A Fast Fission Monte Carlo Code, AEC report BNL-647 (T-206), 1961

```
      DIMENSION SP(10),SBE1(20),SBI1(20),SBF1(20),SBC1(20),SBE2(20),SBI2
     1(20),SBF2(20),SBC2(20),VBOUND(20),AP1(10,20),AP2(10,20),SBL1(20,20
     2),SBL2(20,20),P(22),VL1(20),VL2(20),SL(20),PL(20),FP1(22),FP2(22),
     3LEAK(100,10),NELS(100,10),NINS(100),NFIS(100),KAPT(100)
      READ 1,XS,YS,ZS,PARA,PARB,PARC,THETA,KS,NEUT
1     FORMAT(7F8.4,I2,I14)
      PRINT 2,XS,YS,ZS,PARA,PARB,PARC,THETA,KS,NEUT
2     FORMAT(1H1,3HXS=F8.4,2X,3HYS=F8.4,2X,3HZS=F8.4,2X,5HPARA=F8.4,2X,5
     1HPARB=F8.4,2X,5HPARC=F8.4,2X,6HTHETA=F8.4,2X,3HKS=I2,2X,5HNEUT=I14
     2)
      READ 3,SP
3     FORMAT(10F7.4)
      PRINT 4,SP
4     FORMAT(1H0,3HSP=10F7.4)
      READ 5,XMAX,YMAX,ZMAX,RMAX,KAS
5     FORMAT(4F8.4,I2)
      PRINT 6,XMAX,YMAX,ZMAX,RMAX,KAS
6     FORMAT(1H0,5HXMAX=F8.4,2X,5HYMAX=F8.4,2X,5HZMAX=F8.4,2X,5HRMAX=F8.
     14,2X,4HKAS=I2)
      READ 7,TD,TCH,EMIN,ECH,KT1,KT2
7     FORMAT(4F7.3,2I3)
      PRINT 8,TD,TCH,EMIN,ECH,KT1,KT2
8     FORMAT(1H0,3HTD=F7.3,2X,4HTCH=F7.3,2X,5HEMIN=F7.3,2X,4HECH=F7.3,2X
     1,4HKT1=I3,2X,4HKT2=I3)
      READ 9,P
9     FORMAT(11F6.2)
      PRINT 10,P
10    FORMAT(1H0,2HP=11F6.2/3X,11F6.2)
      READ 11,VBOUND
11    FORMAT(10F7.4)
      PRINT 12,VBOUND
```

```
   12   FORMAT(1H0,7HVBOUND=10F7.4/8X,10F7.4)
        READ 13,AD1,A1,ALIM1,SLIM1,CIN1,VST1,FNU1,DELNU1,KIA1
   13   FORMAT(F7.5,2F7.2,5F8.4,I2)
        PRINT 14,AD1,A1,ALIM1,SLIM1,CIN1,VST1,FNU1,DELNU1,KIA1
   14   FORMAT(1H0,4HAD1=F7.5,2X,3HA1=F7.2,2X,6HALIM1=F7.2,2X,6HSLIM1=F8.4
       1,2X,5HCIN1=F8.4,2X,5HVST1=F8.4,2X,5HFNU1=F8.4,2X,7HDELNU1=F8.4,2X,
       25HKIA1=I2)
        READ 15,SBE1
   15   FORMAT(10F7.3)
        DO 16 J=1,20
   16   SBE1(J)=AD1*SBE1(J)
        PRINT 17,SBE1
   17   FORMAT(1H0,4HSBE=10F7.3/5X,10F7.3)
        READ 15,SBI1
        DO 18 J=1,20
   18   SBI1(J)=AD1*SBI1(J)
        PRINT 19,SBI1
   19   FORMAT(1H0,4HSBI=10F7.3/5X,10F7.3)
        READ 15,SBF1
        DO 20 J=1,20
   20   SBF1(J)=AD1*SBF1(J)
        PRINT 21,SBF1
   21   FORMAT(1H0,4HSBF=10F7.3/5X,10F7.3)
        READ 15,SBC1
        DO 22 J=1,20
   22   SBC1(J)=AD1*SBC1(J)
        PRINT 23,SBC1
   23   FORMAT(1H0,4HSBC=10F7.3/5X,10F7.3)
        READ 15,AP1
        PRINT 24,AP1
   24   FORMAT(1H0,3HAP=10F7.3/4X,10F7.3/4X,10F7.3/4X,10F7.3/4X,10F7.3/4X,
```

```
     110F7.3/4X,10F7.3/4X,10F7.3/4X,10F7.3/4X,10F7.3/4X,10F7.3/4X,10F7.3
     2/4X,10F7.3/4X,10F7.3/4X,10F7.3/4X,10F7.3/4X,10F7.3/4X,10F7.3/4X,10
     3F7.3/4X,10F7.3)
      READ 15,VL1
      PRINT 25,VL1
25    FORMAT(1H0,3HVL=10F7.3/4X,10F7.3)
      READ 15,SBL1
      PRINT 26,SBL1
26    FORMAT(1H0,4HSBL=10F7.3/(5X,10F7.3))
      READ 27,FP1
27    FORMAT(11F6.3)
      PRINT 28,FP1
28    FORMAT(1H0,3HFP=11F6.3/4X,11F6.3)
      READ 13,AD2,A2,ALIM2,SLIM2,CIN2,VST2,FNU2,DELNU2,KIA2
      PRINT 29,AD2,A2,ALIM2,SLIM2,CIN2,VST2,FNU2,DELNU2,KIA2
29    FORMAT(1H4,4HAD2=F7.5,2X,3HA2=F7.2,2X,6HALIM2=F7.2,2X,6HSLIM2=F8.4
     1,2X,5HCIN2=F8.4,2X,5HVST2=F8.4,2X,5HFNU2=F8.4,2X,7HDELNU2=F8.4,2X,
     25HKIA2=I2)
      IF(AD2)40,40,30
30    READ 15,SBE2
      DO 31 J=1,20
31    SBE2(J)=AD2*SBE2(J)
      PRINT 17,SBE2
      READ 15,SBI2
      DO 32 J=1,20
32    SBI2(J)=AD2*SBI2(J)
      PRINT 19,SBI2
      READ 15,SBF2
      DO 33 J=1,20
33    SBF2(J)=AD2*SBF2(J)
      PRINT 21,SBF2
```

```
      READ 15,SBC2
      DO 34 J=1,20
34    SBC2(J)=AD2*SBC2(J)
      PRINT 23,SBC2
      READ 15,AP2
      PRINT 24,AP2
      READ 15,VL2
      PRINT 25,VL2
      READ 15,SBL2
      PRINT 26,SBL2
      READ 27,FP2
      PRINT 28,FP2
      GO TO 50
40    DO 41 J=1,20
41    SBE2(J)=0.0
42    DO 43 J=1,20
43    SBI2(J)=0.0
44    DO 45 J=1,20
45    SBF2(J)=0.0
46    DO 47 J=1,20
47    SBC2(J)=0.0
50    REWIND KT1
      REWIND KT2
      KT=KT1
      MULT=1
      NL=0
      NC=0
      NS=0
      NT=0
      NF=0
      NLTD=0
```

```
      NGTR=0
      NGZR=0
      NLME=0
      NGER=0
      NOSL=0
      KSCAT=0
100   DO 801 N=1,NEUT
110   CALL SOURCE(ALPHA,BETA,GAMMA,VEL,X,Y,Z,TIME,PARA,PARB,PARC,XS,YS,Z
     1S,ZMAX,THETA,SP,KS)
120   CALL SIGMA(VEL,SBE1,SBE2,SBI1,SBI2,SBF1,SBF2,SBC1,SBC2,AD1,AD2,VBO
     1UND,TMFP,PE1,PE2,PI1,PI2,PF1,PF2,PC1,J)
      IF(J)122,122,127
122   NT=NT+1
      IF(NT-5)110,110,124
124   PRINT 125,NT
125   FORMAT(1H0,3HNT=I2)
      GO TO 900
127   NT=0
130   CALL FLIGHT(DIST,TIME,TMFP,VEL)
      IF(DIST)132,135,135
132   PRINT 133,DIST
133   FORMAT(1H0,5HDIST=12E,4)
      GO TO 900
135   IF(TIME)136,140,140
136   PRINT 137,TIME
137   FORMAT(1H0,5HTIME=12E,4)
      GO TO 900
140   GO TO (145,150,155),KAS
145   CALL BLOCK(X,Y,Z,ALPHA,BETA,GAMMA,DIST,KGEO,XMAX,YMAX,ZMAX)
      GO TO (160,600),KGEO
150   CALL CYLIND(X,Y,Z,ALPHA,BETA,GAMMA,DIST,KGEO,RMAX,ZMAX)
```

```
      GO TO (160,600),KGEO
  155 CALL SPHERE(X,Y,Z,ALPHA,BETA,GAMMA,DIST,KGEO,RMAX)
      GO TO (160,600),KGEO
  160 IF(AD2)161,161,165
  161 CALL COLID1(PE1,PI1,PF1,KCOL)
      GO TO 170
  165 CALL COLID2(PE1,PE2,PI1,PI2,PF1,PF2,PC1,KCOL)
  170 KTYPE=KCOL/10
      KNUCL=KCOL-(10*KTYPE)
      GO TO (200,300,400,500),KTYPE
  200 CALL ELTAL(TIME,TD,TCH,Z,ZMAX,KELS,NELS)
      NS=NS+1
      GO TO (203,205,207,207,209),KELS
  203 NLTD=NLTD+1
      GO TO 209
  205 NGTR=NGTR+1
      GO TO 800
  207 NGZR=NGZR+1
      GO TO 800
  209 KSCAT=KSCAT+1
      IF(KSCAT-100)211,211,225
  211 GO TO (215,220),KNUCL
  215 CALL ELSCAT(ALPHA,BETA,GAMMA,VEL,A1,ALIM1,SLIM1,AP1,J)
      GO TO 120
  220 CALL ELSCAT(ALPHA,BETA,GAMMA,VEL,A2,ALIM2,SLIM2,AP2,J)
      GO TO 120
  225 NOSL=NOSL+1
      KSCAT=0
      GO TO 800
  300 CALL INTAL(TIME,TD,TCH,KINS,NINS)
      NS=NS+1
```

```
      GO TO (303,305,305,305,307),KINS
303   NLTD=NLTD+1
      GO TO 307
305   NGTR=NGTR+1
      GO TO 800
307   KSCAT=KSCAT+1
      IF(KSCAT-100)309,309,320
309   GO TO (310,315),KNUCL
310   CALL LEVEL(VEL,SBL1,VBOUND,PL,J)
      CALL INSCAT(ALPHA,BETA,GAMMA,VEL,A1,CIN1,P,PL,VL1,VST1,KIA1)
      GO TO 120
315   CALL LEVEL(VEL,SBL2,VBOUND,PL,J)
      CALL INSCAT(ALPHA,BETA,GAMMA,VEL,A2,CIN2,P,PL,VL2,VST2,KIA2)
      GO TO 120
320   NOSL=NOSL+1
      KSCAT=0
      GO TO 800
400   CALL FISTAL(TIME,TD,TCH,KFIS,NFIS)
      KSCAT=0
      GO TO (402,404,404,404,406),KFIS
402   NLTD=NLTD+1
      GO TO 406
404   NGTR=NGTR+1
      GO TO 800
406   GO TO (407,409),KNUCL
407   CALL FISSN(X,Y,Z,VEL,TIME,FP1,FNU1,DELNU1,NF,KT)
      GO TO 800
409   CALL FISSN(X,Y,Z,VEL,TIME,FP2,FNU2,DELNU2,NF,KT)
      GO TO 800
500   CALL CAPTAL(TIME,TD,TCH,KCAP,KAPT)
      NC=NC+1
```

```
      KSCAT=0
      GO TO (504,506,506,506,507),KCAP
  504 NLTD=NLTD+1
      GO TO 800
  506 NGTR=NGTR+1
  507 GO TO 800
  600 CALL LEKTAL(TIME,VEL,TD,TCH,EMIN,ECH,KLEK,LEAK)
      NL=NL+1
      KSCAT=0
      GO TO (604,606,608,610,611),KLEK
  604 NLTD=NLTD+1
      GO TO 800
  606 NGTR=NGTR+1
      GO TO 800
  608 NLME=NLME+1
      GO TO 800
  610 NGER=NGER+1
  611 GO TO 800
  800 GO TO (801,809,820),MULT
  801 CONTINUE
      KS=1
  803 MULT=2
      REWIND KT1
      REWIND KT2
      IF(NF)850,850,807
  807 N=NF
      NF=0
  809 N=N-1
      IF(N)814,811,811
  811 READ TAPE KT1,XS,YS,ZS,PARA,THETA
      KT=KT2
```

```
      GO TO 110
  814 MULT=3
      REWIND KT1
      REWIND KT2
      IF(NF)850,850,818
  818 N=NF
      NF=0
  820 N=N-1
      IF(N)803,822,822
  822 READ TAPE KT2,XS,YS,ZS,PARA,THETA
      KT=KT1
      GO TO 110
  850 PRINT 851,NL,NC,NS,NF,NLTD,NGTR,NGZR,NLME,NGER,NOSL
  851 FORMAT(1H1,3HNL=I8,2X,3HNC=I8,2X,3HNS=I8,2X,3HNF=I8/1H0,5HNLTD=I8,
     12X,5HNGTR=I8,2X,5HNGZR=I8,2X,5HNLME=I8,2X,5HNGER=I8,2X,5HNOSL=I8)
      PRINT 853,LEAK
  853 FORMAT(1H0,5HLEAK=20I6/(6X,20I6))
      PRINT 855,NELS
  855 FORMAT(1H4,5HNELS=20I6/(6X,20I6))
      PRINT 857,NINS
  857 FORMAT(1H4,5HNINS=20I6/(6X,20I6))
      PRINT 859,NFIS
  859 FORMAT(1H4,5HNFIS=20I6/(6X,20I6))
      PRINT 861,KAPT
  861 FORMAT(1H4,5HKAPT=20I6/(6X,20I6))
      PUNCH 863,LEAK,NELS,NINS,NFIS,KAPT
  863 FORMAT(10I6)
  900 CALL EXIT
      END
```

```
      SUBROUTINE SOURCE(ALPHA,BETA,GAMMA,VEL,X,Y,Z,TIME,PARA,PARB,PARC,X
     1S,YS,ZS,ZMAX,THETA,SP,KS)
      DIMENSION SP(10)
      GO TO (10,20,30,40),KS
10    X=XS
      Y=YS
      Z=ZS
      GAMMAC=2.0*RANNOF(W)-1.0
      VEL=PARA
      CALL ISOANG(ALPHA,BETA,GAMMA,GAMMAC,VEL)
      TIME=THETA
      RETURN
20    X=XS*(2.0*RANNOF(V)-1.0)
      Y=YS*(2.0*RANNOF(W)-1.0)
      Z=ZS
      GAMMA=1.0
      ALPHA=0.0
      BETA=0.0
      VEL=PARA-PARB*RANNOF(U)
      TIME=0.0
      RETURN
30    CALL ANGLS(SP,GAMMAC)
      VEL=PARA-PARB*RANNOF(V)-PARC*(1.0-GAMMAC)
      CALL ISOANG(ALPHA,BETA,GAMMA,GAMMAC,VEL)
      S=(-ZMAX-ZS)/GAMMA
      X=S*ALPHA
      Y=S*BETA
      Z=-ZMAX
      TIME=S/VEL
      RETURN
40    CALL TARGET(ALPHA,BETA,GAMMA,VEL,X,Y,Z,TIME,PARA,PARB,PARC)
```

```
      RETURN
      END
      SUBROUTINE ANGLS(SP,GAMMAC)
      DIMENSION SP(10)
      R=RANNOF(X)
      M=10.0*R+1.0
      REM=R-0.1*FLOATF(M-1)
      IF(10-M)30,10,20
   10 GAMMAC=SP(10)+(REM/0.1)*(1.0-SP(10))
      RETURN
   20 GAMMAC=SP(M)+(REM/0.1)*(SP(M+1)-SP(M))
      RETURN
   30 GAMMAC=1.0
      RETURN
      END
      SUBROUTINE TARGET(ALPHA,BETA,GAMMA,VEL,X,Y,Z,TIME,PARA,PARB,PARC)
      X=X
      Y=Y
      Z=Z
      ALPHA=ALPHA
      BETA=BETA
      GAMMA=GAMMA
      TIME=0.0
      VEL=PARA-PARB*RANNOF(V)-PARC*ABSF(GAMMA)
      RETURN
      END
      SUBROUTINE SIGMA(EN,SBE1,SBE2,SBI1,SBI2,SBF1,SBF2,SBC1,SBC2,AD1,AD
     12,EBOUND,TMFP,PE1,PE2,PI1,PI2,PF1,PF2,PC1,J)
      DIMENSION SBE1(20),SBE2(20),SBI1(20),SBI2(20),SBF1(20),SBF2(20),SB
     1C1(20),SBC2(20),EBOUND(20)
   10 CALL GROUP(EN,EBOUND,J,KGP)
```

```
      J=J
   11 GO TO(12,14),KGP
   12 J=0
   13 RETURN
14    IF(20-J)60,60,20
   20 SE1=FIND(EN,J,EBOUND,SBE1)
   21 SI1=FIND(EN,J,EBOUND,SBI1)
   22 SF1=FIND(EN,J,EBOUND,SBF1)
   23 SC1=FIND(EN,J,EBOUND,SBC1)
   24 IF(AD2)25,25,30
   25 SE2=0.
   26 SI2=0.
   27 SF2=0.
   28 SC2=0.
   29 GO TO 40
   30 SE2=FIND(EN,J,EBOUND,SBE2)
   31 SI2=FIND(EN,J,EBOUND,SBI2)
   32 SF2=FIND(EN,J,EBOUND,SBF2)
   33 SC2=FIND(EN,J,EBOUND,SBC2)
   40 TMFP=1.0/(SE1+SI1+SF1+SC1+SE2+SI2+SF2+SC2)
   41 PE1=TMFP*SE1
   42 PI1=TMFP*SI1
   43 PF1=TMFP*SF1
   44 IF(AD2)45,45,50
   45 PC1=1.0-PE1-PI1-PF1
   46 IF(PC1-0.0001)47,48,48
   47 PC1=0.0
   48 RETURN
   50 PC1=TMFP*SC1
   51 PE2=TMFP*SE2
   52 PI2=TMFP*SI2
```

```
   53 PF2=TMFP*SF2
   54 RETURN
   60    SE1=SBE1(20)
   61    SI1=SBI1(20)
   62    SF1=SBF1(20)
   63    SC1=SBC1(20)
   64    SE2=SBE2(20)
   65    SI2=SBI2(20)
   66    SF2=SBF2(20)
   67    SC2=SBC2(20)
   68    GO TO 40
         END
         SUBROUTINE GROUP(EN,EBOUND,J,KGP)
         DIMENSION EBOUND(20)
   10    IF(EN-EBOUND(1))11,13,13
   11    KGP=1
   12    RETURN
   13    J=20
   14    IF(EN-EBOUND(J))15,91,91
   15    J=10
   16    IF(EN-EBOUND(J))17,91,29
   17    J=5
   18    IF(EN-EBOUND(J))19,91,25
   19    J=2
   20    IF(EN-EBOUND(J))90,91,21
   21    J=J+1
   22    IF(EN-EBOUND(J))90,91,23
   23    J=J+1
   24    IF(EN-EBOUND(J))90,91,91
   25    J=7
   26    IF(EN-EBOUND(J))27,91,21
```

```
27    J=J-1
28    GO TO 24
29    J=15
30    IF(EN-EBOUND(J))31,91,33
31    J=12
32    IF(EN-EBOUND(J))27,91,21
33    J=17
34    IF(EN-EBOUND(J))27,91,21
90    J=J-1
91    KGP=2
92    RETURN
      END
      FUNCTION FIND(EN,J,EBOUND,SBX)
      DIMENSION EBOUND(20),SBX(20)
      FIND=SBX(J)+(EN-EBOUND(J))*(SBX(J+1)-SBX(J))/(EBOUND(J+1)-EBOUND(J
     1))
      RETURN
      END
      SUBROUTINE FLIGHT(DIST,TIME,TMFP,VEL)
   10 B=RANNOF(X)
   12 IF(B-.0000454)10,10,13
   13 C=LOGF(B)
      DIST=TMFP*(-C)
      TIME=TIME+DIST/VEL
      RETURN
      END
      SUBROUTINE BLOCK(X,Y,Z,ALPHA,BETA,GAMMA,DIST,KGEO,XMAX,YMAX,ZMAX)
10    X=X+ALPHA*DIST
11    Y=Y+BETA*DIST
12    Z=Z+GAMMA*DIST
13    IF(ABSF(X)-XMAX)14,14,30
```

```
14    IF(ABSF(Y)-YMAX)15,15,30
15    IF(ABSF(Z)-ZMAX)20,20,30
20    KGEO=1
21    GO TO 50
30    KGEO=2
50    RETURN
      END
      SUBROUTINE SPHERE(X,Y,Z,ALPHA,BETA,GAMMA,DIST,KGEO,RMAX)
10    X=X+ALPHA*DIST
11    Y=Y+BETA*DIST
12    Z=Z+GAMMA*DIST
13    IF(X**2+Y**2+Z**2-RMAX**2)14,14,30
14    KGEO=1
15    GO TO 50
30    KGEO=2
50    RETURN
      END
      SUBROUTINE CYLIND(X,Y,Z,ALPHA,BETA,GAMMA,DIST,KGEO,RMAX,ZMAX)
10    X=X+ALPHA*DIST
11    Y=Y+BETA*DIST
12    Z=Z+GAMMA*DIST
13    IF(ABSF(Z)-ZMAX)14,14,30
14    IF(X**2+Y**2-RMAX**2)20,20,30
20    KGEO=1
21    GO TO 50
30    KGEO=2
50    RETURN
      END
      SUBROUTINE LEKTAL(TIME,VEL,TD,TCH,EMIN,ECH,KLEK,LEAK)
9     DIMENSION LEAK(100,10)
10    ITIME=(TIME-TD)/TCH
```

```
   11   IF(ITIME-1)12,14,14
   12   KLEK=1
   13   RETURN
   14   IF(100-ITIME)15,17,17
   15   KLEK=2
   16   RETURN
   17   IEN=(0.5227*(VEL**2)-EMIN)/ECH
   18   IF(IEN-1)19,21,21
   19   KLEK=3
   20   RETURN
   21   IF(10-IEN)22,24,24
   22   KLEK=4
   23   RETURN
   24   LEAK(ITIME,IEN)=LEAK(ITIME,IEN)+1
   25   KLEK=5
   26   RETURN
        END
        SUBROUTINE COLID1(PE1,PI1,PF1,KCOL)
   9    R=RANNOF(X)
   10   IF(R-PE1)20,11,11
   11   IF(R-PE1-PI1)30,12,12
   12   IF(R-PE1-PI1-PF1)40,13,13
   13   KCOL=41
   14   RETURN
   20   KCOL=11
   21   RETURN
   30   KCOL=21
   31   RETURN
   40   KCOL=31
   41   RETURN
        END
```

```
      SUBROUTINE COLID2(PE1,PE2,PI1,PI2,PF1,PF2,PC1,KCOL)
    9 R=RANNOF(X)
   10 IF(R-PE1)20,11,11
   11 IF(R-PE1-PE2)30,12,12
   12 IF(R-PE1-PE2-PI1)40,13,13
   13 IF(R-PE1-PE2-PI1-PI2)50,14,14
   14 IF(R-PE1-PE2-PI1-PI2-PF1)60,15,15
   15 IF(R-PE1-PE2-PI1-PI2-PF1-PF2)70,16,16
   16 IF(R-PE1-PE2-PI1-PI2-PF1-PF2-PC1)80,90,90
   20 KCOL=11
   21 RETURN
   30 KCOL=12
   31 RETURN
   40 KCOL=21
   41 RETURN
   50 KCOL=22
   51 RETURN
   60 KCOL=31
   61 RETURN
   70 KCOL=32
   71 RETURN
   80 KCOL=41
   81 RETURN
   90 KCOL=42
   91 RETURN
      END
      SUBROUTINE ELTAL(TIME,TD,TCH,Z,ZMAX,KELS,NELS)
      DIMENSION NELS(100,10)
   10 ITIME=(TIME-TD)/TCH
   11 IF(ITIME-1)12,14,14
   12 KELS=1
```

```
13    RETURN
14    IF(100-ITIME)15,17,17
15    KELS=2
16    RETURN
17    IZ=6.0+(5.0*Z)/ZMAX
18    IF(IZ-1)19,21,21
19    KELS=3
20    RETURN
21    IF(10-IZ)22,24,24
22    KELS=4
23    RETURN
24    KELS=5
25    NELS(ITIME,IZ)=NELS(ITIME,IZ)+1
26    RETURN
      END
      SUBROUTINE ELSCAT(ALPHA,BETA,GAMMA,VEL,A,ALIM,SLIM,AP,J)
      DIMENSION AP(10,20)
10    IF(VEL-SLIM)11,20,20
11    GAMMAC=2.0*RANNOF(X)-1.0
12    IF(A-ALIM)13,15,15
13    CALL CMLAB(ALPHA,BETA,GAMMA,GAMMAC,VEL)
14    RETURN
15    CALL ISOANG(ALPHA,BETA,GAMMA,GAMMAC,VEL)
16    RETURN
20    CALL ANGLE(J,AP,GAMMAC)
21    GO TO 12
      END
      SUBROUTINE ANGLE(J,AP,GAMMAC)
      DIMENSION AP(10,20)
      R=RANNOF(X)
      M=10.0*R+1.0
```

```
      REM=R-0.1*FLOATF(M-1)
      IF(10-M)30,10,20
   10 GAMMAC=AP(10,J)+(REM/0.1)*(1.0-AP(10,J))
      RETURN
   20 GAMMAC=AP(M,J)+(REM/0.1)*(AP(M+1,J)-AP(M,J))
      RETURN
   30 GAMMAC=1.0
      RETURN
      END
      SUBROUTINE CMLAB(ALPHA,BETA,GAMMA,GAMMAC,VEL,A)
10    R1=RANNOF(X)
11    R2=RANNOF(X)
12    ETA=(2.0*R1-1.0)**2+(2.0*R2-1.0)**2
13    IF(ETA-1.0)14,14,10
14    ROOT=SQRTF((1.0-GAMMAC**2)/ETA)
15    ALPHAC=(2.0*R1-1.0)*ROOT
16    BETAC=(2.0*R2-1.0)*ROOT
17    RTG=SQRTF(1.0-GAMMA**2)
18    ALPHAP=((ALPHA*GAMMA*ALPHAC-BETA*BETAC)/RTG)+ALPHA*GAMMAC
19    BETAP=((BETA*GAMMA*ALPHAC+ALPHA*BETAC)/RTG)+BETA*GAMMAC
20    GAMMAP=-ALPHAC*RTG+GAMMA*GAMMAC
21    RTA=SQRTF(1.0+A**2+2.0*A*GAMMAC)
22    ALPHA=(ALPHA+A*ALPHAP)/RTA
23    BETA=(BETA+A*BETAP)/RTA
24    GAMMA=(GAMMA+A*GAMMAP)/RTA
25    VEL=(VEL*RTA)/(A+1.0)
26    RETURN
      END
      SUBROUTINE ISOANG(ALPHA,BETA,GAMMA,GAMMAC,VEL)
10    GAMMA=GAMMAC
11    R1=RANNOF(X)
```

```
12    R2=RANNOF(X)
13    ETA=(2.0*R1-1.0)**2+(2.0*R2-1.0)**2
14    IF(ETA-1.0)15,15,11
15    ROOT=SQRTF((1.0-GAMMA**2)/ETA)
16    ALPHA=(2.0*R1-1.0)*ROOT
17    BETA=(2.0*R2-1.0)*ROOT
18    VEL=VEL
19    RETURN
      END
      SUBROUTINE INTAL(TIME,TD,TCH,KINS,NINS)
9     DIMENSION NINS(100)
10    ITIME=(TIME-TD)/TCH
11    IF(ITIME-1)12,14,14
12    KINS=1
13    RETURN
14    IF(100-ITIME)15,17,17
15    KINS=2
16    RETURN
17    NINS(ITIME)=NINS(ITIME)+1
18    KINS=5
19    RETURN
      END
      SUBROUTINE LEVEL(VEL,SBL,VBOUND,PL,J)
      DIMENSION SBL(20,20),VBOUND(20),PL(20),SL(20)
      IF(20-J)10,10,20
10    DO 15 L=1,20
      SL(L)=SBL(L,20)
15    CONTINUE
      GO TO 30
20    DO 25 L=1,20
      SL(L)=SBL(L,J)+(VEL-VBOUND(J))*(SBL(L,J+1)-SBL(L,J))/(VBOUND(J+1)-
```

```
     1VBOUND(J))
  25 CONTINUE
     SUM=0.0
     DO 30 L=1,20
     SUM=SUM+SL(L)
  30 CONTINUE
     SUMI=1.0/SUM
     DO 35 L=1,20
     PL(L)=SUMI*SL(L)
  35 CONTINUE
     RETURN
     END
     SUBROUTINE INSCAT(ALPHA,BETA,GAMMA,VEL,A,CIN,P,PL,VL,VST,KIA)
     DIMENSION PL(20),VL(20),P(22)
  10 GO TO (11,14),KIA
  11 GAMMAC=2.0*RANNOF(X)-1.0
  12 CALL ISOANG(ALPHA,BETA,GAMMA,GAMMAC,VEL)
  13 GO TO 20
  14 CALL ANGLI(VEL,A,GAMMAC)
  15 GO TO 12
  20 IF(VEL-VST)21,30,30
  21 R1=RANNOF(X)
  22 L=1
  23 SUM=0.
  24 SUM=SUM+PL(L)
  25 IF(R1-SUM)28,26,26
  26 L=L+1
  27 GO TO 24
  28 VEL=VEL-VL(L)
  29 RETURN
  30 CALL INSPEC(VEL,CIN,P)
```

```
31    RETURN
      END
      SUBROUTINE ANGLI(VEL,A,GAMMAC)
      GAMMAC=1.0
      VEL=VEL
      A=A
      RETURN
      END
      SUBROUTINE INSPEC(VEL,CIN,P)
      DIMENSION P(22)
      VMAX=CIN*VEL
      R=RANNOF(X)
      K=20.0*R+1.0
      REM=R-0.05*FLOATF(K-1)
      W=P(K)+(REM/0.05)*(P(K+1)-P(K))
      VEL=W*VMAX
      RETURN
      END
      SUBROUTINE CAPTAL(TIME,TD,TCH,KCAP,KAPT)
9     DIMENSION KAPT(100)
10    ITIME=(TIME-TD)/TCH
11    IF(ITIME-1)12,14,14
12    KCAP=1
13    RETURN
14    IF(100-ITIME)15,17,17
15    KCAP=2
16    RETURN
17    KAPT(ITIME)=KAPT(ITIME)+1
18    KCAP=5
19    RETURN
      END
```

```
      SUBROUTINE FISTAL(TIME,TD,TCH,KFIS,NFIS)
    9 DIMENSION NFIS(100)
   10 ITIME=(TIME-TD)/TCH
   11 IF(ITIME-1)12,14,14
   12 KFIS=1
   13 RETURN
   14 IF(100-ITIME)15,17,17
   15 KFIS=2
   16 RETURN
   17 NFIS(ITIME)=NFIS(ITIME)+1
   18 KFIS=5
   19 RETURN
      END
      SUBROUTINE FISSN(X,Y,Z,VEL,TIME,FP,FNU,DELNU,NF,KT)
      DIMENSION FP(22)
      FISNO=FNU+DELNU*(VEL**2)
      IF(FISNO-3.0)20,30,40
   20 R1=RANNOF(W)+2.0
       IF(R1-FISNO)30,30,25
   25 I=2
      GO TO 50
   30 I=3
      GO TO 50
   40 IF(FISNO-4.0)41,49,49
   41 R2=RANNOF(W)+3.0
       IF(R2-FISNO)49,49,45
   45 I=3
      GO TO 50
   49 I=4
   50 DO 60 N=1,I
      R3=RANNOF(W)
```

```
      K=20.0*R3+1.0
      REM=R3-0.05*FLOATF(K-1)
      PARA=FP(K)+(REM/0.05)*(FP(K+1)-FP(K))
      THETA=TIME
      XS=X
      YS=Y
      ZS=Z
      WRITE TAPE KT,XS,YS,ZS,PARA,THETA
   60 NF=NF+1
      RETURN
      END
```