To the Graduate Council:

I am submitting herewith a dissertation written by Michael E. Dunn entitled "Development of a Continuous Energy Version of the Monte Carlo Code KENO V.a." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirement for the degree of Doctor of Philosophy, with a major in Nuclear Engineering.

H. L. Dodds, Major Professor

We have read this thesis
and recommended its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of the Graduate School

Development of a Continuous Energy Version of the Monte Carlo Code KENO V.a

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Michael E. Dunn

December 1996

Acknowledgements

I would like to thank my advisor, Dr. H. L. Dodds, for his technical guidance and overall supervision of this research. Appreciation is also expressed to Dr. L. M. Petrie and Dr. D. F. Hollenbach for their technical advice and guidance throughout this work. I would also like to thank my committee members Dr. M. Guidry, Dr. L. F. Miller and Dr. R. E. Pevey for their many contributions. I wish to thank the members, past and present, of the Nuclear Criticality Safety Group at the University of Tennessee, Knoxville; specifically, B. Basoglu, C. Bentley, S. Goluoglu, and L. Paschal. Finally, I would like to thank Dr. Bob Busch for providing additional benchmark problems.

The research was performed under appointment to the Nuclear Engineering/Health Physics Fellowship program administered by Oak Ridge Institute for Science and Education under the contract number DE-AC05-76OR00033 between the U.S. Department of Energy and Oak Ridge Associated Universities.

Abstract

KENO V.a is a multigroup Monte Carlo code which solves the Boltzmann Transport Equation and is used extensively in the Nuclear Criticality Safety community to calculate the effective multiplication factor (k_{eff}) of systems containing fissile material. Due to the smaller amount of disk storage and cpu time required in calculations, multigroup approaches have been preferred over continuous energy (point) approaches in the past for solving transport problems. With the advent of high performance computers, storage and cpu limitations are less restrictive, thereby making continuous energy methods viable for transport calculations. Moreover, continuous energy methods avoid many of the assumptions and approximations inherent in multigroup methods. Since a continuous energy version of KENO V.a does not exist, the objective of the work is to develop a new version of KENO V.a which utilizes continuous energy cross sections. Currently, a point cross section library which is based on a raw continuous energy cross section library such as ENDF/B-V is not available for implementation in KENO V.a; however, point cross section libraries are available for MCNP, another widely used Monte Carlo transport code. Since MCNP cross sections are based on ENDF data and are readily available, a new version of KENO V.a named PKENO V.a has been developed which performs the random walk using MCNP cross sections. In order to utilize point cross sections, extensive modifications have been made to KENO V.a. In order to test the new code, PKENO V.a was benchmarked against the multigroup version of KENO V.a, MCNP and experiment using fifty test problems. The calculated results obtained with PKENO are in agreement (i.e., within $\pm$ 2σ) with both

MCNP, KENO V.a and experiment. As a result, PKENO can be used to perform Nuclear Criticality Safety Evaluations.

Table of Contents

Chapter **Page**

1. Introduction 1
    1.1 Background 1
    1.2 Previous Work 4
    1.3 Objectives 8
    1.4 Importance of Research 8
    1.5 Originality of Work 10
    1.6 Scope and Organization 12

2. Multigroup KENO V.a 14
    2.1 Geometry Package 15
    2.2 Multigroup Cross Sections 18
    2.3 Program Flow 21
    2.4 Particle Tracking 33

3. Continuous Energy KENO V.a 42
    3.1 Input and Data Handling 42
    3.2 Cross Section Processing 44
    3.3 Continuous Energy Random Walk 45
        3.3.1 Source Distribution 45
        3.3.2 Collision Site Selection 46
        3.3.3 Collision Treatment 48
    3.4 Program Flow 63

4. Results 73
    4.1 KENO V.a Benchmark Problems 73
    4.2 MCNP Benchmark Problems 77
    4.3 Additional Benchmark Problems 81
    4.4 Calculational Efficiency 84

5. Conclusions and Recommendations 90
    5.1 Conclusions 90
    5.2 Recommendations for Future Work 94

References 96

Appendix A: Listing of PKENO V.a Subroutine TRACK 100

Appendix B: Listing of PKENO V.a Subroutine COLLISION 124

Appendix C: PKENO V.a Input Listing for Benchmark Problems 153

Appendix D: KENO V.a Input Listing for Benchmark Problems 182

Appendix E: MCNP Input Listing for Benchmark Problems 212

Appendix F: Instructions for Executing PKENO V.a 268

Vita 272

List of Figures

Figure 2.1: Example of Stacking Units in KENO V.a.[13] 15
Figure 2.2: Geometrical Configurations Not Allowed in KENO.[13] 16
Figure 2.3: Example of Hole Option in KENO V.a.[13] 17
Figure 2.4: KENO V.a Program Initialization Flow Diagram.[13] 22
Figure 2.5: Flow Diagram for Subroutine MASTER.[13] 24
Figure 2.6: Flow Diagram for Subroutine DATAIN.[13] 27
Figure 2.7: Flow Diagram for Subroutine GUIDE.[13] 30
Figure 2.8: Flow Diagram for Subroutine TRACK.[13] 32
Figure 2.9: Particle Tracking Flow Diagram for Subroutine TRACK.[13] 35
Figure 2.10: Collision Treatment Flow Diagram for Multigroup Version of KENO V.a. 38
Figure 3.1: Collision Treatment Flow Diagram for Multigroup Version of KENO V.a. 52
Figure 3.2: Collision Treatment Flow Diagram for Continuous Energy Version of KENO V.a. 53
Figure 3.3: Cross Section Hierarchy for Particle Interactions. 56
Figure 3.4: PKENO V.a Flow Diagram of Subroutine MASTER. 64
Figure 3.5: Flow Diagram for Subroutine RDXSDIR. 66
Figure 3.6: Flow Diagram for Subroutine XSPROC. 68
Figure 3.7: Flow Diagram for Subroutine TRACK in PKENO V.a. 69
Figure 3.8: Flow Diagram for Subroutine COLLISION in PKENO V.a. 72
Figure 4.1: Diagram of SHEBA II Critical Assembly. 84

List of Tables

Table 1.1: Different Monte Carlo Codes Used in Criticality Safety Community. 5
Table 2.1: Example of Probability Array for Group to Group Scattering. 19
Table 2.2: Multigroup Example of Discrete Angle Probability Array. 20
Table 2.3: Multigroup Example of Discrete Angle Array. 20
Table 2.4: Description of Subroutines Called by MASTER[13] 25
Table 2.5: Description of Subroutines Called by DATAIN[13] 28
Table 2.6: Description of Subroutines Called by GUIDE[13] 30
Table 2.7: Description of Subroutines Called by TRACK[13] 33
Table 3.1: Description of Subroutines Called by MASTER in PKENO V.a . . . 65

Table 3.2: Description of Subroutines Called by RDXSDIR. 67
Table 3.3: Description of Subroutines Called by XSPROC. 68
Table 3.4: Description of Subroutines Called by TRACK in PKENO V.a 70
Table 3.5: Description of Subroutines Called by COLLISION in PKENO V.a . . 71
Table 4.1: Description of the 25 KENO V.a Benchmark Problems.[13] 75
Table 4.2: Calculated k_{eff}'s for the 25 KENO V.a Benchmark Problems. 76
Table 4.3: Description of the MCNP Primer Benchmark Problems.[34] 79
Table 4.4: Calculated k_{eff}'s for MCNP Primer Benchmark Problems. 79
Table 4.5: Description of the MCNP Neutron Benchmark Problems.[35] 80
Table 4.6: Calculated k_{eff}'s for MCNP Neutron Benchmark Problems. 80
Table 4.7: Description of Additional Computational Benchmark Problems.[10] . . . 82
Table 4.8: Calculated k_{eff}'s for Additional Computational Benchmark Problems. 83
Table 4.9: Comparison of CPU Times for the 25 KENO V.a Benchmark Problems. 87
Table 4.10: Comparison of CPU Times for MCNP Benchmark Problems. 88
Table 4.11: Comparison of CPU Times for Additional Benchmark Problems. . . 89

Chapter One

Introduction

1.1 Background

The primary objective in transport theory is to solve the Boltzmann transport equation. The two modern methods which are utilized for solving transport problems are discrete ordinates and Monte Carlo. In the discrete ordinates method, a discretized form of the linear Boltzmann transport equation is obtained by integrating the transport equation over a differential phase space cell. As a result, the transport equation is formulated as a finite difference equation which can be solved numerically. The solution of the finite difference equation will approach the exact solution as the space, energy, and angle meshes approach differential size.[1] In contrast to the deterministic discrete ordinates method, Monte Carlo is a stochastic method which solves the Integral or Peierl's form of the Boltzmann transport equation.[2] The Monte Carlo solution is based on random walks of point-to-point transport of individual radiation particles. If sufficient random sampling is performed, the average value obtained is an accurate estimate of the quantity being calculated. Unlike discrete ordinates, Monte Carlo can treat very complex and unusual geometries and permits the geometric reproduction of the physical model in as much detail as necessary.

The Monte Carlo method has been in existence for centuries; however, the method did not gain significant attention until the nuclear weapons development of World

War II.[3,4,5] During the Manhattan Project, Nicholas Metropolis named the method "Monte Carlo" because of the similarity of statistical simulation to games of chance.[3] Also during the 1940's, the Electronic Numerical Integrator and Computer (ENIAC), which is the world's first electronic computer, was built.[4] Since the 1940's and 1950's, the theory and application of Monte Carlo to radiation transport problems has evolved with the development of more advanced computing systems.[6] Initially, the early codes were written in machine language for specific problems.[4] With the standardization of programming languages in the 1960's, codes became more general and portable between computing platforms.

While computer codes were being developed in the 1950s, another evolution was taking place which is relevant to transport calculations. In order to perform any nuclear calculation, the analyst must use the appropriate cross section data. Due to the complicated dependence of cross sections on temperature, incident particle energy and angle, the associated quantity of data can be enormous. Cross sections are typically compiled from experimental measurements and theoretical nuclear models.[4,7] The earliest record of cross section tabulation is the single volume report BNL-325, which is commonly called the "barn-book," and was published in 1955. By the mid 1960s, the second edition of BNL-325 was published and encompassed six volumes. Since the task of measuring and compiling cross sections is so laborious, the nuclear industry within the United States decided to standardize and consolidate the cross section data into a single data set.[7] As a result, analysts from the national laboratories and commercial power industry joined to create the Evaluated Nuclear Data File (ENDF) system.[4] The

philosophy of the ENDF system is to provide evaluated cross sections in a retrievable format for nuclear applications.[8] There are two different evaluated libraries which are maintained in the ENDF system. The ENDF/A set contains both incomplete and complete sets of cross sections. In contrast, the ENDF/B set contains only one recommended evaluation for each nuclide, and the evaluation is as complete as possible.[7,8] The ENDF/B data set is continually updated and revised as new information is acquired. The latest version of evaluated cross section data is ENDF/B-VI.[8,9] It should be noted that there are comparable data sets in the world such as the UKNDL and JEF cross sections in the United Kingdom and Europe, respectively.[7,10]

Typically, Monte Carlo codes are classified according to their cross section treatment. In particular, a code can solve the transport equation using either continuous energy or multigroup cross sections; however, some codes are versatile and possess both capabilities. Cross sections are typically measured and constructed in a pointwise or continuous energy format. In an evaluated data set such as ENDF, thousands of points might be used to represent a nuclide's cross section from 20 MeV down to 1.0e-5 eV. Since early computing systems could not handle the huge amount of data, the codes were developed to utilize multigroup cross sections. Group dependent cross sections are generated from the continuous energy data set by averaging the data with respect to a flux spectrum over a specified number of energy groups.[11] Over the years, multigroup cross section processing has become a specialized and well developed field within the nuclear industry. As a result, multigroup cross section sets generally represent the true structure of continuous energy cross sections in sufficient detail. In recent years, the

increased storage and CPU speed of computing systems has permitted the use of continuous energy cross sections for transport calculations.

1.2 Previous Work

Although the early nuclear weapons program provided the initial catalyst for Monte Carlo methods development, several different and independent Monte Carlo codes have been developed over the past fifty years. These independent efforts can be attributed to the various national labs competing with differing research applications.[12] In addition, many of the labs acquired different and incompatible computing systems which contributed further to independent code development; however, the standardization of programming languages has somewhat alleviated the problem with incompatible computing platforms.

In addition to independent code development, the differences in cross section processing over the years has further contributed to the differences in transport codes. Due to the voluminous nature and complexity of an evaluated cross section data set, transport codes do not access the raw data directly. Instead, cross section processing codes are used to process the raw data into a working library which is more amenable to the particular transport code. Then, the working library can be used with a particular code to perform a transport calculation. Consequently, the development of independent Monte Carlo codes with different cross section requirements has lead to the development of several different working libraries in different formats.

Despite the more unified efforts in recent years, several different Monte Carlo codes are used in the Nuclear Criticality Safety (NCS) community. The more prominent codes which are used to perform criticality safety analyses are listed in Table 1.1.

Table 1.1: Different Monte Carlo Codes Used in Criticality Safety Community.

Code	Author	Type†
KENO V.a[13,14]	Oak Ridge National Laboratory (ORNL)	criticality
MCNP 4a[4]	Los Alamos National Laboratory (LANL)	general purpose
MONK 6[10]	United Kingdom Atomic Energy Authority (UKAEA)	general purpose
RACER[12]	Knolls Atomic Power Laboratory (KAPL)	general purpose
VIM[15]	Argonne National Laboratory (ANL)	general purpose

Each of the different Monte Carlo codes has three common computational modules: geometry, physics, and editing.[16] Despite the basic similarities, each code listed in Table 1 has distinct features and characteristics which makes each code unique.

KENO V.a uses multigroup cross sections and was developed specifically for criticality safety analyses.[10,14] KENO is used as part of the SCALE (Standardized Computer Analysis for Licensing Evaluation) code system which is used to perform standardized computer analysis for licensing evaluations.[14,17] The geometry package in KENO is markedly different from other Monte Carlo codes. Simple body structures such as cuboids (i.e., rectangular parallelepiped), cylinders, spheres, hemispheres and

†A criticality code only performs an eigenvalue calculation for the system. A general purpose code can perform either a fixed source or an eigenvalue calculation.

hemicylinders are used to model system configurations.[13] Complicated configurations are modeled by combining or stacking the simple geometric units. KENO's geometry package permits a user to model a complex system rather quickly with minimal input. The ability to model complex systems with simple geometric configurations usually reduces the amount of CPU time for calculations without a loss of accuracy. As a result, KENO has a distinct advantage over other codes with regard to speed and accuracy in modeling complex systems.[10]

MCNP is a code that can be used for neutron, photon, electron or coupled neutron/photon/electron transport.[4] MCNP can perform either a fixed source or eigenvalue calculation using either continuous energy or multigroup cross sections. The continuous energy libraries were obtained by processing the raw ENDF (Evaluated Nuclear Data File) data using the NJOY[18] code system. A distinct advantage with MCNP is that continuous energy libraries based on ENDF/B-V and -VI data are available for calculations. With regard to geometry, MCNP has a generalized geometry package which uses logical combinations of surfaces (e.g., spheres, cylinders, planes, etc.) to model complex systems.[4,10] Because of the broad scope of the code and large storage requirements for the libraries, the code requires a substantial amount of overhead for execution which can be a disadvantage for users with limited computing capabilities.

MONK is a Monte Carlo code which is primarily used to solve eigenvalue problems.[10,17] The code performs transport calculations using either continuous energy or multigroup cross sections. The continuous energy library was derived from the UKNDL and JEF cross section libraries which are used in the United Kingdom and

Europe. MONK also uses simple body structures to model fissile system configurations. In addition to the simple structures, the geometry package uses holes which can be constructed using structures such as square or hexagonal arrays, plates or a helix screw.[10] In contrast with KENO V.a, the simple structures can be rotated and permitted to overlap giving MONK more geometrical versatility. With regard to MCNP, MONK has a similar computer overhead disadvantage required to execute the code.[10]

RACER is a general purpose, neutronic Monte Carlo code which originated from the MCV[21] code developed at Knolls Atomic Power Laboratory.[19,20] In addition to the eigenvalue capability, the code can also solve complex fixed source problems.[19] The cross section treatment in RACER is rather unique. In particular, a continuous energy treatment is used for neutrons in the epithermal range (i.e., above 0.625 eV) while a 32 group treatment is used for neutrons in the thermal range.[19,21] With regard to the geometry package, three dimensional regions are modeled using combinations of surfaces.[20]

VIM uses either continuous energy or multigroup cross sections to solve neutron and/or photon transport problems.[22] The code can be used to solve either criticality or fixed source problems. The geometrical options include infinite medium, combinatorial† geometry, plane lattice, and rectangular or hexagonal lattices of combinatorial geometry cells.[15] VIM has been used extensively at ANL for the past twenty years for the simulation of zero power critical assemblies.[23]

---

†In combinatorial geometry, three dimensional configurations are modeled by considering unions, differences and intersections of simple bodies such as spheres, boxes, cylinders, etc.[2]

1.3 Objectives

The objective of this work is to develop and evaluate a new version of KENO V.a which utilizes continuous energy cross sections. Currently, a point cross section library which is based on a raw continuous energy cross section library such as ENDF/B-V is not available for implementation in KENO V.a; however, point cross section libraries are available for MCNP.[4] Since MCNP cross sections are based on ENDF data and are readily available, a new version of KENO V.a named PKENO V.a which performs the random walk using MCNP cross sections is developed. By utilizing the same cross section libraries, a better comparison can be made between the two codes because the differences in the results reside in the methodology and not in cross section data.

PKENO V.a is benchmarked against the multigroup version of KENO V.a and MCNP using the standard 25 KENO test problem set.[13] In addition, the code is tested against 25 additional benchmark problems which represent a wide range of critical systems. In each case, PKENO is compared with KENO V.a, MCNP and experiment, when applicable.

1.4 Importance of Research

Operation and storage limits for fissile material are provided in standards developed by the American National Standards Institute (ANSI). Typically, these limits are overly restrictive for certain operation and storage conditions. In order to circumvent this issue, the standards permit the establishment of subcritical limits based upon calculations which have been shown to be valid when compared with applicable

experimental data.[24] However, there is an ever increasing number of nontraditional processes which involve unique and complex configurations of fissile material (e.g., weapons dismantling, radioactive waste, etc.), and experimental data which replicates the actual configuration of these unique processes is not available. Consequently, a particular code that has been validated against conventional experimental data may be used to establish subcritical limits for a unique system outside of the code's range of applicability. From a computational standpoint, it is crucial for an analyst to use a code which can model the neutron physics and geometry of a system as faithfully as possible. Typically, the Monte Carlo method is used to analyze complex systems because it has the distinct advantage of being able to solve directly (i.e., without approximation) the integral form of the Boltzmann Transport Equation as well as handle complicated geometries which are impossible with deterministic codes.[1,25] In addition, the Monte Carlo method can use either multigroup or continuous energy cross sections to solve the transport equation.

Multigroup cross section sets are generated from point cross section data by averaging over a specified number of energy groups. If the fine energy detail of the cross sections is needed for a particular problem, a broad multigroup library may not be sufficient to analyze the system. Since a flux weighting spectrum is used to generate a multigroup library, the multigroup cross section set is only valid for a system having a flux spectrum similar to the corresponding weighting spectrum. In contrast, point cross sections provide sufficient detail of the cross section energy dependence for most problems, and a single continuous energy library can be used for a wide variety of

problems. Since angle and energy are treated as continuous variables in point Monte Carlo, exact energy angle formulas can be used to describe the kinematics of particle collisions. As a result, the continuous energy random walk is more analogous to the actual particle transport. Since point Monte Carlo closely resembles the true stochastic transport, the method can be used to evaluate the accuracy of multigroup cross section libraries and other calculational methods. In the past, multigroup approaches have been preferred over continuous energy approaches due to the smaller amount of disk storage and cpu time required for multigroup calculations. With the advent of high performance computers, storage and cpu limitations are less restrictive, thereby making continuous energy methods viable for transport calculations.

KENO V.a is a multigroup Monte Carlo code which is the most widely used production level criticality safety code in the nuclear industry.[26,27] Moreover, the code is used extensively in the nuclear community within the United States to perform standardized computer analyses of nuclear systems for licensing evaluations.[17] A continuous energy version of KENO V.a would be extremely beneficial to the world-wide nuclear community because many of the assumptions and approximations inherent in the multigroup approach are avoided.

1.5 Originality of Work

The research is original because a continuous energy version of KENO V.a does not exist. Although a significant amount of work has been performed in the area of continuous energy Monte Carlo, the codes which utilize continuous energy cross sections

are significantly different from KENO V.a. In particular, KENO is the only code written specifically for performing criticality safety analysis. Furthermore, MCNP, RACER and VIM all use generalized geometry to describe three dimensional systems whereas KENO uses simple body structures which can be stacked or combined to model a system configuration. Since MONK also uses simple body structures, it is the only code which closely resembles KENO from a geometrical standpoint. In contrast with KENO, MONK's continuous energy library is based on the UKNDL and JEF libraries, not ENDF data. Moreover, MONK's neutron tracking procedure is different from KENO. In a standard Monte Carlo random walk, the mean free path varies with respect to material boundaries. The distance to the next collision site is randomly selected using the total macroscopic cross section for the material. If the particle traverses a material boundary without a collision and enters a new material, the total cross section for the new material is used to select the collision site in the new region. MONK uses standard Monte Carlo tracking for simple body structures; however, when a hole† is encountered Woodcock†† tracking is used to determine the next collision site.[10] As a result, the boundary tracking in the complex region is replaced by the task of determining a constant mean free path for the entire "hole."[10] Unlike MONK, KENO uses standard boundary tracking throughout the random walk.

---

†A "hole," as defined in MONK, is a complex geometry region which can be a square or hexagonal array, plate, helix screw, etc.[10]

††In Woodcock tracking, a constant mean free path is determined for the entire "hole" based on the different materials present.

This work is also original with respect to the proposed cross section implementation. As noted in the Background Section, each continuous energy code uses a working library which is generated from the original evaluated data set. Since working libraries are generated for a particular code, two different Monte Carlo codes will usually have dissimilar working libraries even though the working libraries are generated from the same evaluated data set. However, in this work, PKENO V.a utilizes the MCNP cross sections which are based on ENDF data. Consequently, the random walk in both codes is performed using the same data set. Although both codes utilize the same cross sections, the random walks employed in PKENO and MCNP are different due to the fundamental differences of each code (i.e, geometry package, fission treatment, etc.). A review of the open literature indicates that production level Monte Carlo codes which utilize the same continuous energy working library simply do not exist.

1.6 Scope and Organization

This dissertation assumes a basic understanding of Monte Carlo and transport theory. Consequently, a derivation of the Boltzmann Transport equation as solved by KENO V.a is not provided. The reader may consult the KENO V.a Manual[13] for a comprehensive derivation of the transport equation. This dissertation is composed of five chapters. The first chapter provides background information related to Monte Carlo codes and continuous energy cross sections. In addition, a survey of previous work and the objectives of the research are presented. The second chapter provides a brief overview of KENO V.a which includes a description of the geometry package,

multigroup cross sections and particle tracking.

Chapter Three provides a detailed description of the development of the continuous energy version of KENO V.a (i.e., PKENO V.a). The results of the PKENO V.a benchmark study are presented in Chapter Four. Finally, conclusions and recommendations for future work are presented in Chapter Five.

Chapter Two

Multigroup KENO V.a

In order to understand the development of the continuous energy version of KENO which is presented in Chapter Three, it is imperative to review the multigroup version of KENO. KENO V.a is a multigroup Monte Carlo code which was developed at the Oak Ridge National Laboratory (ORNL). KENO is a relatively large code consisting of 175 subroutines. The code is predominately written in FORTRAN 77 with a few C subroutines. The primary objective of KENO V.a is to calculate the eigenvalue (i.e., k_{eff}) of the transport equation. The effective multiplication factor is defined using either of the following definitions:

$$k_{eff} = \frac{number\ of\ neutrons\ in\ the\ (i+1)\ generation}{number\ of\ neutrons\ in\ the\ i^{th}\ generation} \qquad (2.1)$$

$$k_{eff} = \frac{rate\ of\ neutron\ production}{rate\ of\ neutron\ loss}. \qquad (2.2)$$

In addition, other calculated quantities such as lifetime, generation time, energy dependent leakages, energy and region dependent absorptions, fissions, fluxes, and fission densities can be calculated.

2.1 Geometry Package

One of the distinct advantages of Monte Carlo over deterministic methods is the ability to model complex geometrical configurations. The KENO V.a geometry package is much simpler than the surface and zone specifications of generalized geometry codes such as MCNP; however, the geometry package of KENO is extremely versatile. The geometry package was adopted from the British code GEM and is based on simple shapes which include spheres, cylinders, hemicylinders, hemispheres and cuboids (i.e., rectangular parallelpipeds).[2] These shapes are assembled together to form a unit, and units are stacked together to form an array or lattice. The array can be placed in another unit using the ARRAY or CORE option, and the process is repeated until the desired geometrical modeled is completed. A generic example of stacking units to form an array is presented in Figure 2.1.

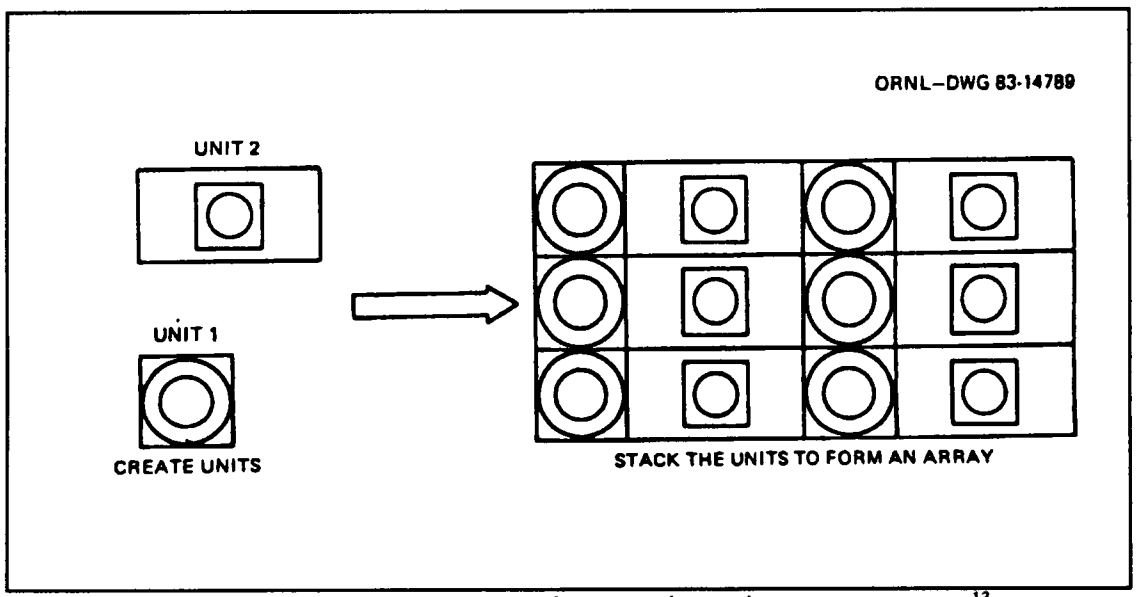


Figure 2.1: Example of Stacking Units in KENO V.a.[13]

Units are constructed by enclosing one region by another region (e.g., sphere in a cuboid). An exterior region must completely enclose and not intersect with an interior region; however, the two regions can be tangent at a particular plane. Each unit has its own orthogonal coordinate system; however, each coordinate system must have the same orientation. Although each unit has a unique coordinate system, all of the geometrical data in the model are correlated to an absolute coordinate system of a global unit or array. In addition to the nonintersection restriction, geometrical regions cannot be rotated with respect to the defining coordinate system. A schematic diagram of geometrical configurations not allowed by KENO V.a is presented in Figure 2.2.

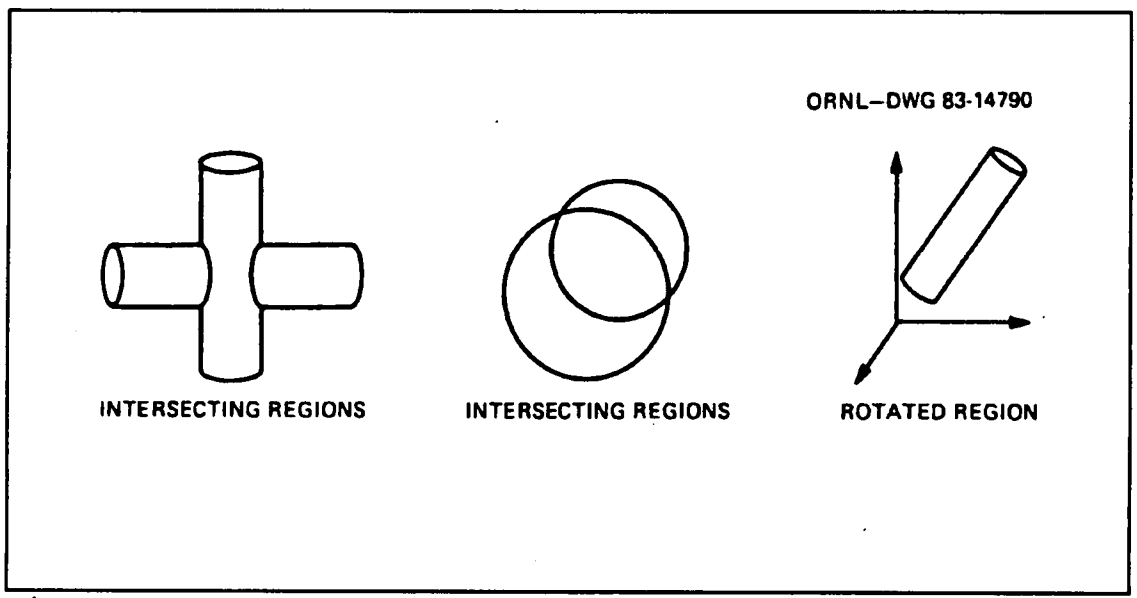


Figure 2.2: Geometrical Configurations Not Allowed in KENO.[13]

As noted above, a geometric region can be placed in a unit as long as it completely encloses all interior regions and does not intersect another region. There are special options known as "extended geometry descriptions" which circumvent the

restriction of complete enclosure. These descriptions include the ARRAY option, which was discussed previously, and the HOLE option. A hole permits the placement of a unit anywhere within a region provided intersections do not occur. The hole cannot be tangent with another hole or a region of the unit containing the hole. An generic example illustrating the HOLE option is presented in Figure 2.3.

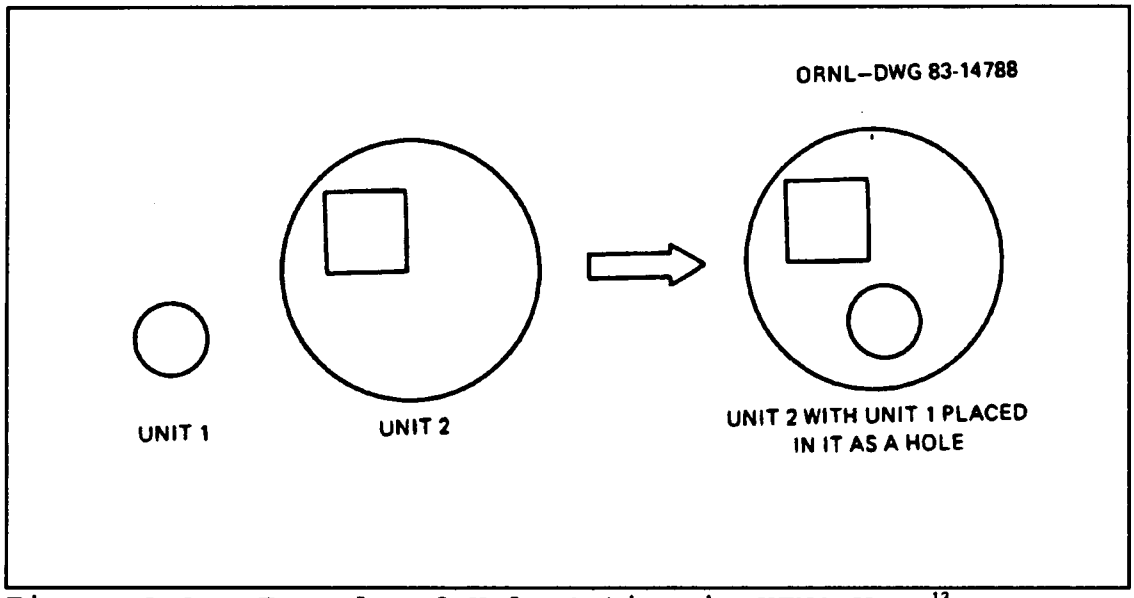


Figure 2.3: Example of Hole Option in KENO V.a.[13]

The nonintersection restriction greatly simplifies the tracking procedure during the random walk. As the particle leaves one region, KENO "knows" which zone it is entering unlike a generalized geometry code which must perform a series of "if-then" tests to determine which cell or zone contains the particle. As a result, particle tracking in the KENO geometry package is extremely efficient.

2.2 Multigroup Cross Sections

KENO V.a reads multigroup cross sections from an ICE mixed library or from an AMPX working library. During the random walk, the code needs cross section information on a per mixture basis (e.g., total cross section for the mixture). If the cross sections are in an AMPX format, KENO generates a mixed cross section library based on the mixing table data input by the user. The mixing table describes the constituents (i.e., nuclide and associated atomic number density) present in each mixture. The one dimensional cross sections used by the code include:

Σ_{tg}^{i} = macroscopic total cross section for mixture i and energy group g.
Σ_{sg}^{i} = macroscopic scattering cross section for mixture i and group g.
Σ_{fg}^{i} = macroscopic fission cross section for mixture i and group g.
ν_g = number of neutrons per fission for energy group g.
χ_g^{i} = fission spectrum for mixture i and energy group g.

Two dimensional cross sections are used to process a collision during the random walk. Angular scattering in KENO V.a is treated by discretizing a P_N Legendre polynomial expansion of the continuous angular variable such that the N+1 moments of the discrete distribution are the same as the truncated polynomial. The discretization yields a set of N+1 equations giving (N+1)/2 discrete polar angles (i.e., angle cosines which are in the lab system) with corresponding (N+1)/2 probabilities.[25] The P_0 term corresponds to the group-to-group energy transfer probability. For a given group-to-group transfer, a new discrete scattering angle is randomly selected based on the associated probabilities. If the scattering is isotropic, the new direction cosine is not selected from a discrete distribution. In particular, the P_0 term determines the appropriate group-to-group transfer for isotropic scattering, and the new direction cosines

are selected from an isotropic distribution instead of a discrete distribution.[13,14] At each collision site, this selection process yields the energy group and angle of the exiting neutron. It should be noted that these discrete angles are laboratory system scattering angles and are not fixed in space. Moreover, this angular treatment should not be confused with the angular quadrature used in discrete ordinate analysis which is fixed in space. The discrete scattering angles will be different for each group to group transfer for each mixture.[25] Examples of a 2-D scattering transfer array, discrete angle probability array and discrete angle array are presented in Tables 2.1, 2.2 and 2.3, respectively. As shown in Table 2.1, the probability for scattering from group 1 to 3 in mixture 1 is 19.3%. For the group 1 to 3 transfer in Tables 2.2 and 2.3, there is a 100% probability the angle will be 0.400109.

Table 2.1: Example of Probability Array for Group to Group Scattering.

To Grp	From Grp 1	From Grp 2	From Grp 3	From Grp 4
+ 0	0.254209	0.310539	0.890619	1.000000
+ 1	0.448609	0.504657	0.109381	
+ 2	0.192706	0.184804		
+ 3	0.104476			

Table 2.2: Multigroup Example of Discrete Angle Probability Array.

To Grp	From Grp 1	From Grp 2	From Grp 3	From Grp 4
+ 0	1.000000	1.000000	1.000000	1.000000
+ 1	1.000000	1.000000	1.000000	
+ 2	1.000000	1.000000		
+ 3	1.000000			

Table 2.3: Multigroup Example of Discrete Angle Array.

To Grp	From Grp 1	From Grp 2	From Grp 3	From Grp 4
+ 0	0.309220	0.556326	0.445350	0.252410
+ 1	0.640449	0.623199	0.301259	
+ 2	0.400109	0.181010		
+ 3	0.126825			

2.3 Program Flow

KENO V.a is a relatively large code consisting of 175 subroutines. The code is predominately written in FORTRAN 77 with a few C subroutines. A comprehensive flow chart of the program cannot be presented due to the voluminous nature of the code. However, to understand the coding modifications discussed in Chapter Three, several abbreviated flow charts will be presented to establish the logical program flow of the original version of KENO V.a. The following description is not an exhaustive description of the program. For more detailed information, the reader is referred to the KENO V.a manual.[13]

KENO is a modular code where each component of the program performs a distinct task or set of tasks. The first portion of the code initializes the program prior to calling subroutine MASTER, which controls the program flow of KENO V.a. The abbreviated flow chart for the program initialization is presented in Figure 2.4. As shown in Figure 2.4 and all subsequent flow diagrams, the names contained in boxes represent subroutines and names without boxes represent calls to library routines. A subroutine with an associated arrow represents a call to another section of the program. As indicated in Figure 2.4, OPNFIL is called to initialize the logical I/O units. Subroutine INITAL performs the initialization of the program and calls PARAM which sets the default parameters and reads the user specified parameter data from the input file. Prior to calling MASTER, subroutine ALOCAT is called to allocate the required amount of storage for the code. Once the program is initialized, MASTER is called to coordinate the logical program flow of KENO. Following a return from MASTER, the

code calls CLOSDA which closes the direct access files and terminates the problem.

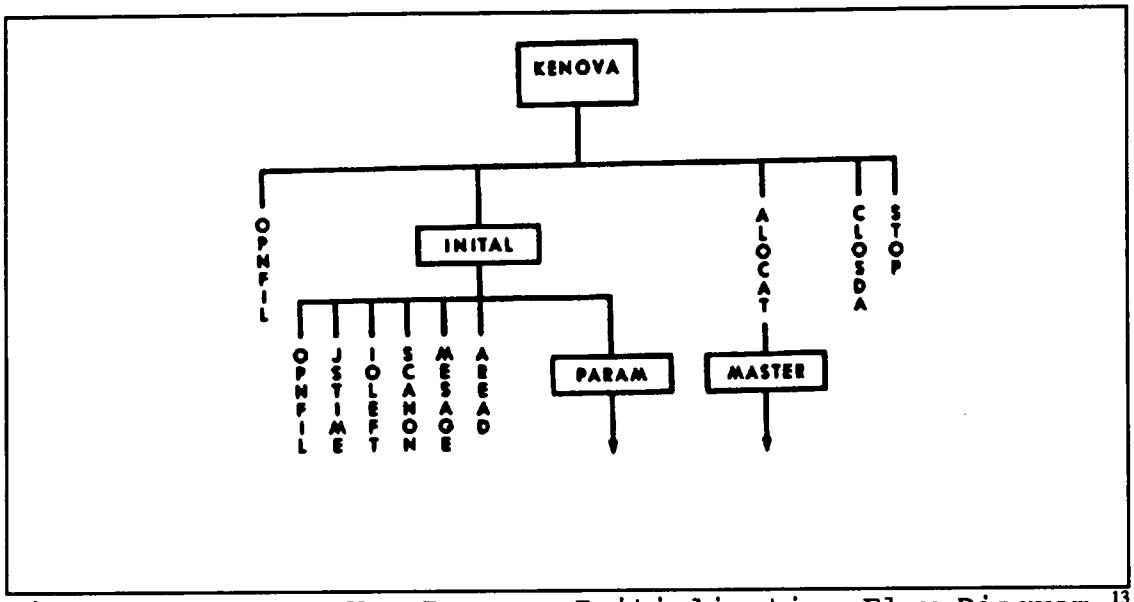


Figure 2.4: KENO V.a Program Initialization Flow Diagram.[13]

The overall program flow of KENO V.a is controlled by subroutine MASTER which is presented in Figure 2.5. As shown in Figure 2.5, MASTER calls several subroutines to perform a variety of tasks. A brief description of each subroutine is summarized in Table 2.4. The reading of all the input data except the title and parameter data is governed by subroutine DATAIN. Based on the input data, cross sections are mixed if mixing table data is read and an ICE mixed library is not provided. If a Monte Carlo formatted mixed cross section library is specified, ICEMIX reads the ICE mixed library. Once the cross sections are read, KENO V.a writes restart information and generates albedo-cross section information if necessary. NSUPG and POINT are called

to process the data into a supergroup format† and calculate pointers, respectively. JOMITY is called to process the geometry information specified in the input file, and PRTPLT prints a two dimensional user defined plot of the geometrical configuration if plot data is entered. Prior to initializing the calculation subroutines, KENO calls CLEAR to initialize the arrays which store calculated data and LODWTS to load energy and region dependent biasing data into memory. Once the data required for the calculation is processed, MASTER calls GUIDE to coordinate the program flow through the Monte Carlo calculation. After the calculation is complete, KEDIT and FITFLX are called to edit the calculated results and load the fluxes for printing, respectively. FREAK is called to generate the frequency distribution for the calculated k_{eff}'s. Prior to returning to subroutine KENOVA, JSTIME is called to determine the total amount of time used by the code.

†If insufficient memory is available to read all of the group dependent data during the calculation, the code separates the information into "supergroups," which can be processed one at a time. If sufficient memory is available, the group dependent data is stored in one supergroup.

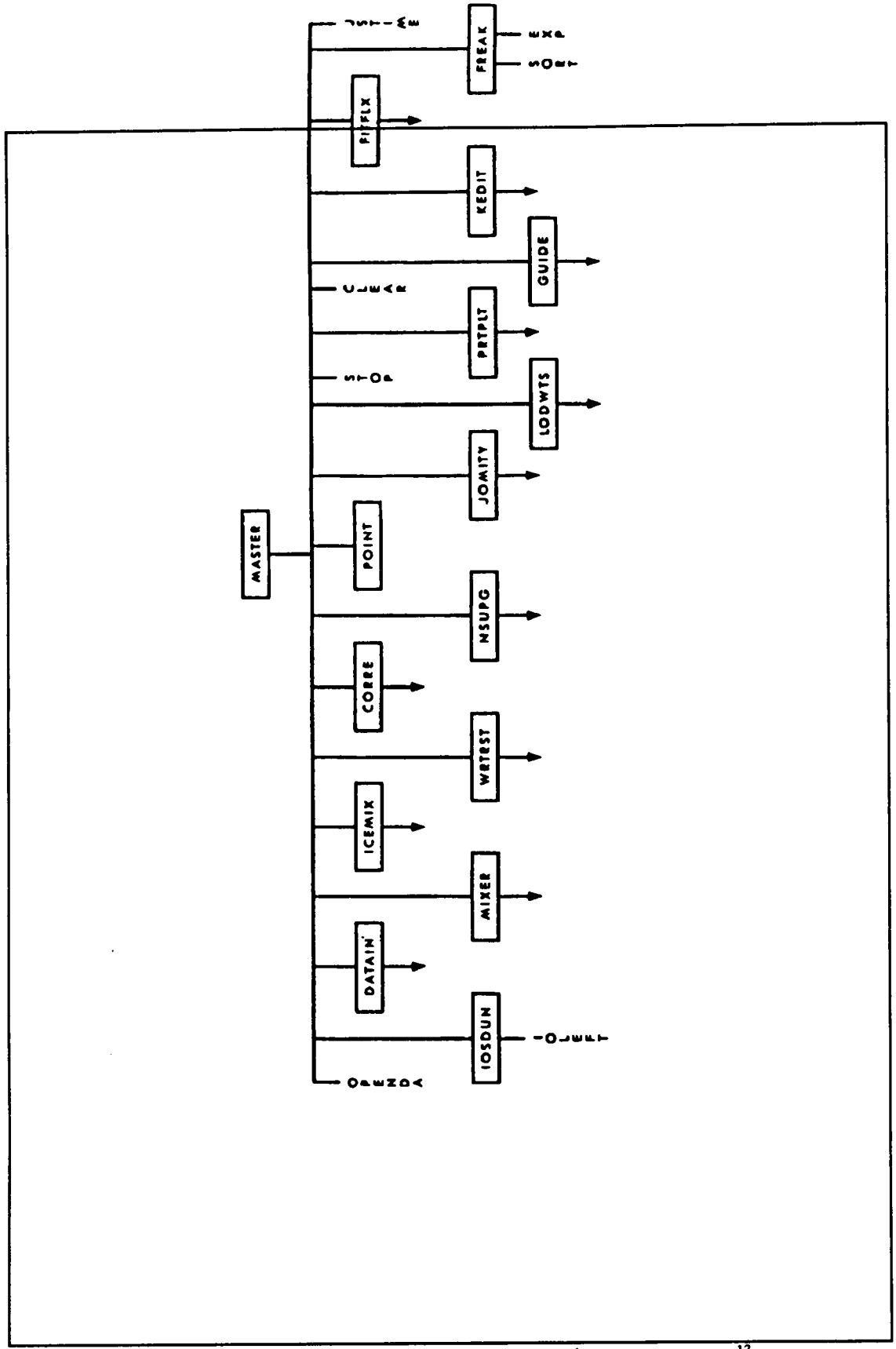


Figure 2.5: Flow Diagram for Subroutine MASTER.[13]

Table 2.4 Description of Subroutines Called by MASTER[13]

Subroutine	Description
OPENDA	open direct access files
IOSDUN	initialize I/O's
DATAIN	read input data
MIXER	mix cross sections
ICEMIX	read ICE mixed library
WRTRST	write restart data
CORRE	generate albedo data
NSUPG	generate supergroup information
POINT	calculate pointers
JOMITY	process geometry information
PRTPLT	print user defined 2-D plots
CLEAR	initialize arrays for calculated data
LODWTS	load biasing data
GUIDE	control tracking
KEDIT	edit calculated data
FITFLX	load fluxes for printing
FREAK	generate k_{eff} frequency distribution
JSTIME	determine time used by code

The modular component of KENO V.a which controls the reading of the input data except for the title and parameter information is DATAIN. An abbreviated flow chart for DATAIN is presented in Figure 2.6, and a description of each subroutine is summarized in Table 2.5. The data specified in the input file can be entered in any order. KENO scans the input for the READ flag, and processes the data accordingly. AREAD is used to identify the READ flag and the associated keyword which identifies the subsequent data in the input file. The possible data include geometry, array, albedo biasing, start, extra 1-D cross section, mixing table and plot data. Depending on which keyword is specified, the appropriate subroutine is called to process the data. For example, READ MIX *mixing table data* END MIX directs DATAIN to call MIXIT to process the mixing table block of data. The subroutines which are responsible for reading the specified input data are identified in Table 2.5.

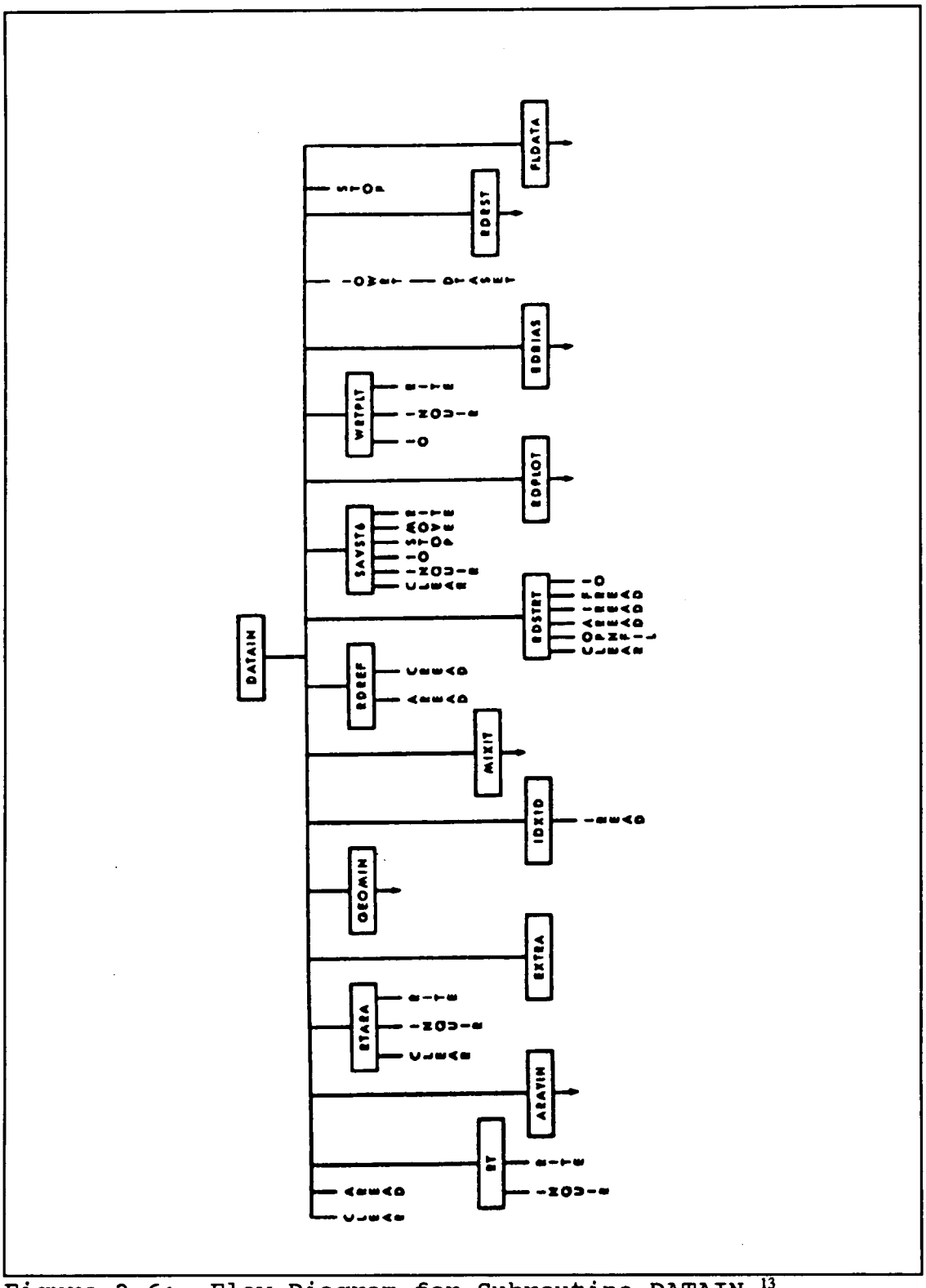


Figure 2.6: Flow Diagram for Subroutine DATAIN.[13]

Table 2.5 Description of Subroutines Called by DATAIN[13]

Subroutine	Description
AREAD	identify READ flag and associated keyword
RT	write data on direct access file
RITE	called by RT to write arrays on direct access file
ARAYIN	read geometrical array size and unit orientation data
RTARA	read array data and write information on direct access
EXTRA	read extra user data not processed by KENO
GEOMIN	read geometry data
IDX1D	read extra 1-D cross sections if entered
MIXIT	reads mixing table data and defines mixtures
RDREF	read boundary conditions
RDSTRT	read start data block
SAVST6	save data start type 6 data
RDPLOT	read the user specified plot information
WRTPLT	loads plot data on direct access
RDBIAS	read biasing data
IOWRT	generate table of unit numbers which specify location of data
DTASET	called by IOWRT to provide data set name of requested I/O unit
RDRST	read restart information from restart data file
FLDATA	supply default data for arrays not specified in input file

Once the input data is processed by DATAIN, control of the program returns to MASTER. The code then processes the various information such as geometry data, supergrouping data, pointers, etc. prior to calling GUIDE which controls the tracking of histories. An abbreviated flow chart for GUIDE is presented in Figure 2.7, and a brief description of the subroutines is provided in Table 2.6. If restart information is provided, RDCALC loads the calculated data (e.g, k_{eff}'s, leakage, absorptions, etc.) from the previous run. Otherwise, START is called to provide the initial source distribution which is region and energy dependent. Once the starting distribution is established, GUIDE performs a loop over the specified number of generations and calculates the neutronic quantities (e.g., k_{eff}, leakage, absorption, etc.) for each generation. Prior to calling TRACK, BANKER places the particles in the current supergroup at the top of the neutron bank and all other particles at the bottom of the bank. After processing the data required for the calculation, GUIDE calls TRACK to perform the tracking of each history through the system. Upon completion of the generation, NSTART accumulates the fission source which is the source for the next generation, and FISFLX calculates the statistics for calculated quantities such as k_{eff}, fission, absorptions, leakages, etc. If restart information is to be saved, WRTCAL and WRTGRP writes the calculated information for the current generation on the restart data file. After the current generation is completed, GUIDE repeats the process until the specified number of generations is complete.

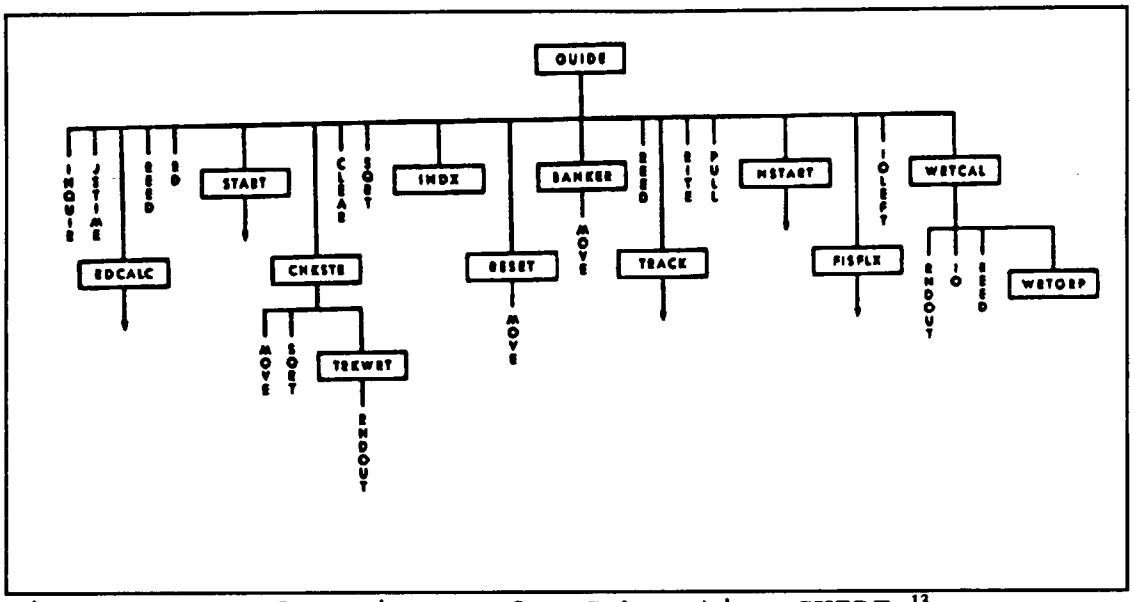


Figure 2.7: Flow Diagram for Subroutine GUIDE.[13]

Table 2.6 Description of Subroutines Called by GUIDE[13]

Subroutine	Description
RDCALC	read data from restart data file
START	provide initial source for problem
CHKSTR	print starting points
INDX	locate cross section index
RESET	count histories in a supergroup
BANKER	sort neutron bank
TRACK	track particle histories
NSTART	provide fission source for next generation
FISFLX	calculate statistics
WRTCAL	write data on restart file

Subroutine TRACK is called by GUIDE to perform tracking of the individual histories. Most of the CPU time required by KENO is devoted to tracking the neutrons through the system. Consequently, TRACK is a very large subroutine which performs many tasks during the random walk. An abbreviated flow diagram for TRACK is provided in Figure 2.8, and a summary of the various subroutines is provided in Table 2.7. If the problem requires albedo information, an initialization call is made to ALBIN. If debug information is requested, LDWRT is called to print debugging information. Prior to processing the history, MOVE is called to load the neutron bank for use in the calculation. MOVE is used several times to move data in and out of storage. TRKWRT is called to print information about the current history. As the history is being processed, FLTRN is used to provide a random number between zero and one, and EXPRN is used to provide the number of mean free paths to the next collision by picking a random number from an exponential distribution. Subroutine CROS is used to determine if the neutron has moved out of one geometry region and entered another region, and LOCBOX determines the unit number for the specified position in an array. If albedo data is used, ALBEDO is called to process a differential albedo reflection by returning the reflected direction cosine for the neutron incident on the reflector material. During the collision treatment, GTISO provides the directions cosines from an isotropic distribution if the collision is isotropic. SFLRA provides a random number between -1.0 and 1.0, SQRT calculates the direction cosines of a history and AZIRN provides the sine and cosine of a random azimuthal angle. When the history escapes from the system or is killed via Russian roulette, the next history is processed. When the specified number

of histories is complete, the next generation is processed.

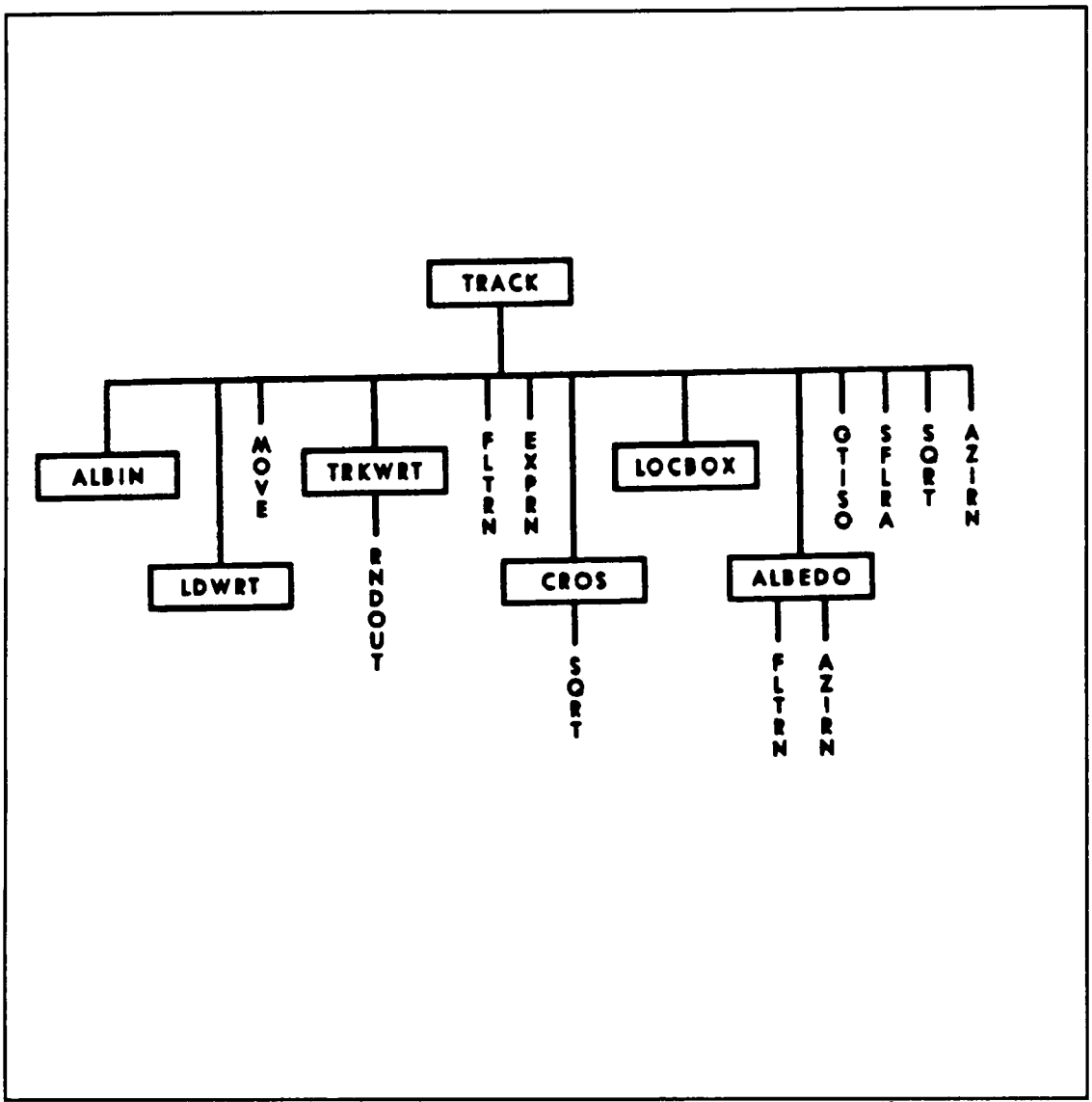


Figure 2.8: Flow Diagram for Subroutine TRACK.[13]

Table 2.7 Description of Subroutines Called by TRACK[13]

Subroutine	Description
ALBIN	initialize albedo information
LDWRT	print debug information
MOVE	move data in and out of storage
TRKWRT	print information about the current neutron
FLTRN	provide random number between zero and one
EXPRN	provide random number from exponential distribution
CROS	processes inward and outward geometry crossing
LOCBOX	determine the current geometrical unit
ALBEDO	process albedo reflection
GTISO	provide isotropic direction cosines
SFLRA	provide random number between -1.0 and 1.0
SQRT	calculate direction cosines
AZIRN	provide sine and cosine of a random azimuthal angle

2.4 Particle Tracking

The following is a brief summary of the particle tracking methodology used in the multigroup version of KENO V.a and is based on the information presented in Ref. 13. As noted in Section 2.3, the random walk is controlled by subroutines GUIDE and TRACK. Two of the most important arrays used in particle tracking are **nubank** and **fsbank**. Both arrays are two dimensional and contain information about each particle such as coordinates, energy group, direction cosines, etc. In addition, particle information is temporarily stored in COMMON /NUTRON/ during the actual processing

of the particle history. Prior to calling TRACK, START is used to generate the initial source distribution. The spatial distribution of the source is selected from either a cosine distribution, uniform volume distribution, a point, user-specified coordinates or from a previous calculation. With respect to the angular component of the source, the initial particle direction is selected from an isotropic distribution. Based on the spatial location of the particle, the initial energy group of each neutron is selected from the fission spectrum for the mixture containing fissile material. The information describing the initial source distribution is stored in **nubank**.

After processing the source, TRACK is called to track each history through the system. In order to understand the logic of particle tracking in KENO V.a, a flow diagram of TRACK is presented in Figure 2.9. The tracking procedure is divided into ten separate sections which include *FSTART*, *PATH*, *INWARD*, *FINDBOX*, *POSIT*, *OUTWARD*, *COLLISION* (*XSEC*), *FISSION*, *ARRAY*, *ALBEDO*.

FSTART initializes the data and sets flags necessary for processing the history. At the beginning of a new history, MOVE is called to load the particle information from **nubank** into COMMON /NUTRON/. If the history is new (i.e. unprocessed), the *PATH* portion of TRACK is executed. If the history is a split neutron, the particle proceeds to *COLLISION* to undergo the collision treatment. If the history resided in the system previously (e.g., a particle which transferred to a new supergroup) and the weight is large enough the particle proceeds to *PATH*. Otherwise, Roulette is played. Upon surviving roulette, the history proceeds to *PATH*.

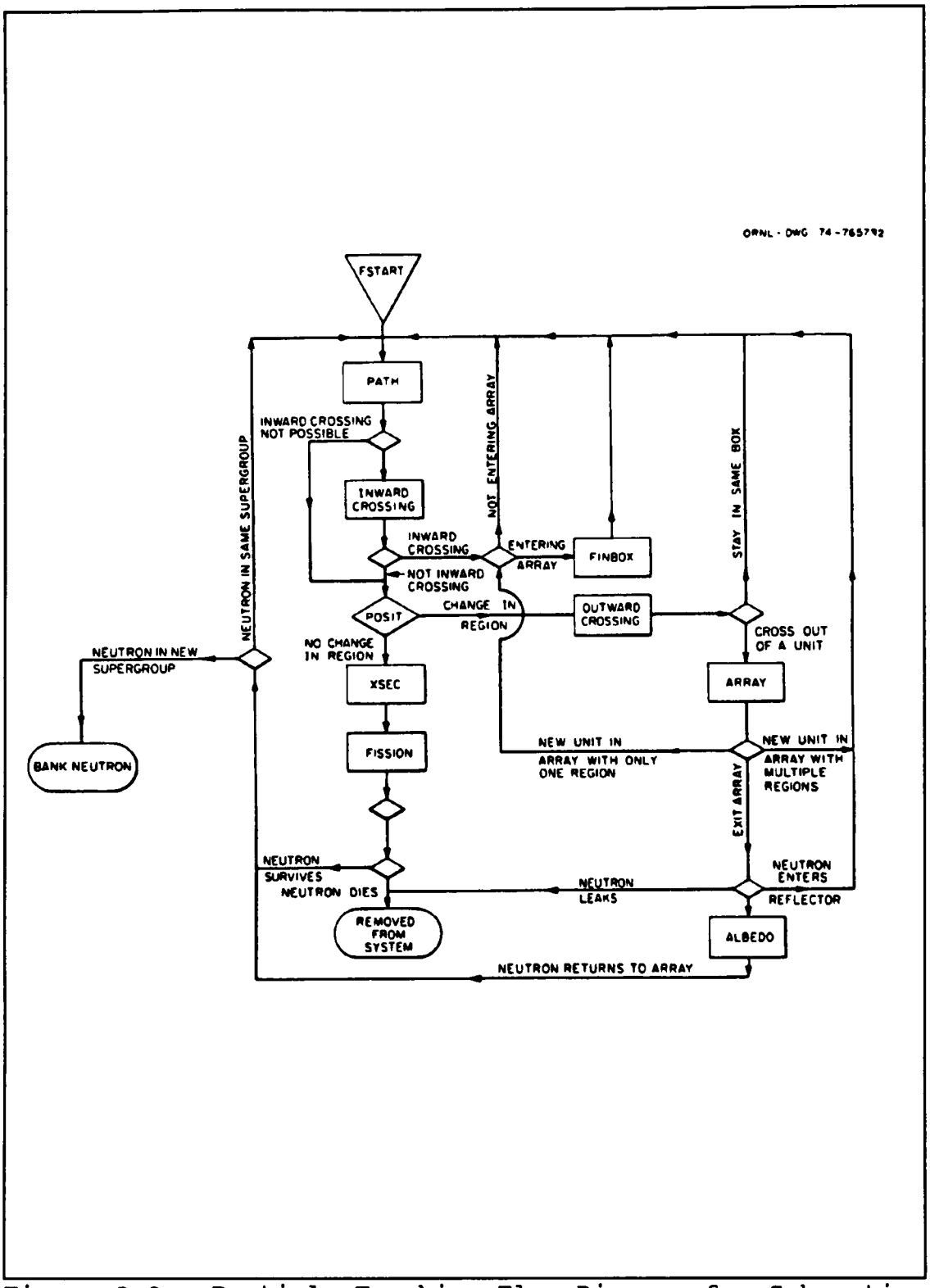


Figure 2.9: Particle Tracking Flow Diagram for Subroutine TRACK.[13]

In the *PATH* section of TRACK, the path length, which is the number of mean free paths the particle will travel in the given mixture, is calculated based on a random selection from an exponential distribution. If the current region is not a void, the distance traveled is the remaining path length divided by the macroscopic total cross section for the mixture; otherwise, the path length is the maximum chord length of the void.

As shown in Figure 2.9, TRACK determines whether an inward crossing is possible. An inward crossing is not possible if the particle is in the innermost region of a unit and the region does not contain holes. If an inward crossing is not possible, the history proceeds to *POSIT*. Otherwise, *INWARD* calls subroutine CROS to determine whether a crossing occurred and the coordinates of the crossing. If a crossing occurred and holes are not present in the region, the flux and neutron age are updated and tracking proceeds to *FINDBOX*. If holes are present, each hole is checked for an inward crossing. If the crossing is into a hole, the coordinates of the particle are transformed to the coordinate system of the hole, and the history proceeds to *FINDBOX*. The final result of an inward crossing is the determination of the correct new region number of the particle. If a particle enters a new array, *FINDBOX* determines the particle's location within the array. The history then proceeds to the *PATH* section of TRACK.

POSIT is used to determine if the collision occurs in the same region which it originated. If there is no change in region, *COLLISION* is used to process the interaction; otherwise, the history proceeds to *OUTWARD*.

The *OUTWARD* section of TRACK uses subroutine MOVE to load the dimensions of the new geometry region into COMMON /NUTRON/. Prior to calling CROS, a flag is set indicating an outward crossing occurred. Subroutine CROS is used to determine the coordinates of the boundary crossing in the system. *OUTWARD* also calculates the number of mean free paths remaining for the history and the particle's contribution to the flux and neutron age. If the particle does not exit the current unit, tracking proceeds to *PATH*. If the history exits a unit in an array, the *ARRAY* portion of TRACK is used to continue tracking the particle. If the problem is a single unit problem or the history exits the external reflector, the *ALBEDO* section is used to process a leakage or albedo reflection.

When a collision occurs, the *COLLISION* portion of TRACK is used to process the collision. A flow diagram of the collision treatment is presented in Figure 2.10. At this point in tracking, the contribution to the flux and age are calculated and the remaining path length is set to zero. In addition, the new weight of the particle, the absorption weight, fission weight, the contribution to the average number of neutrons per fission and the self-multiplication of the unit are calculated. As indicated in Figure 2.10, splitting and Roulette is performed depending on the weight of the particle. A new energy group is selected based on the group-to-group scattering probability for the mixture. If the collision is anisotropic, the cosine of the scattering angle in the lab system is selected based on the associated probabilities for the particular

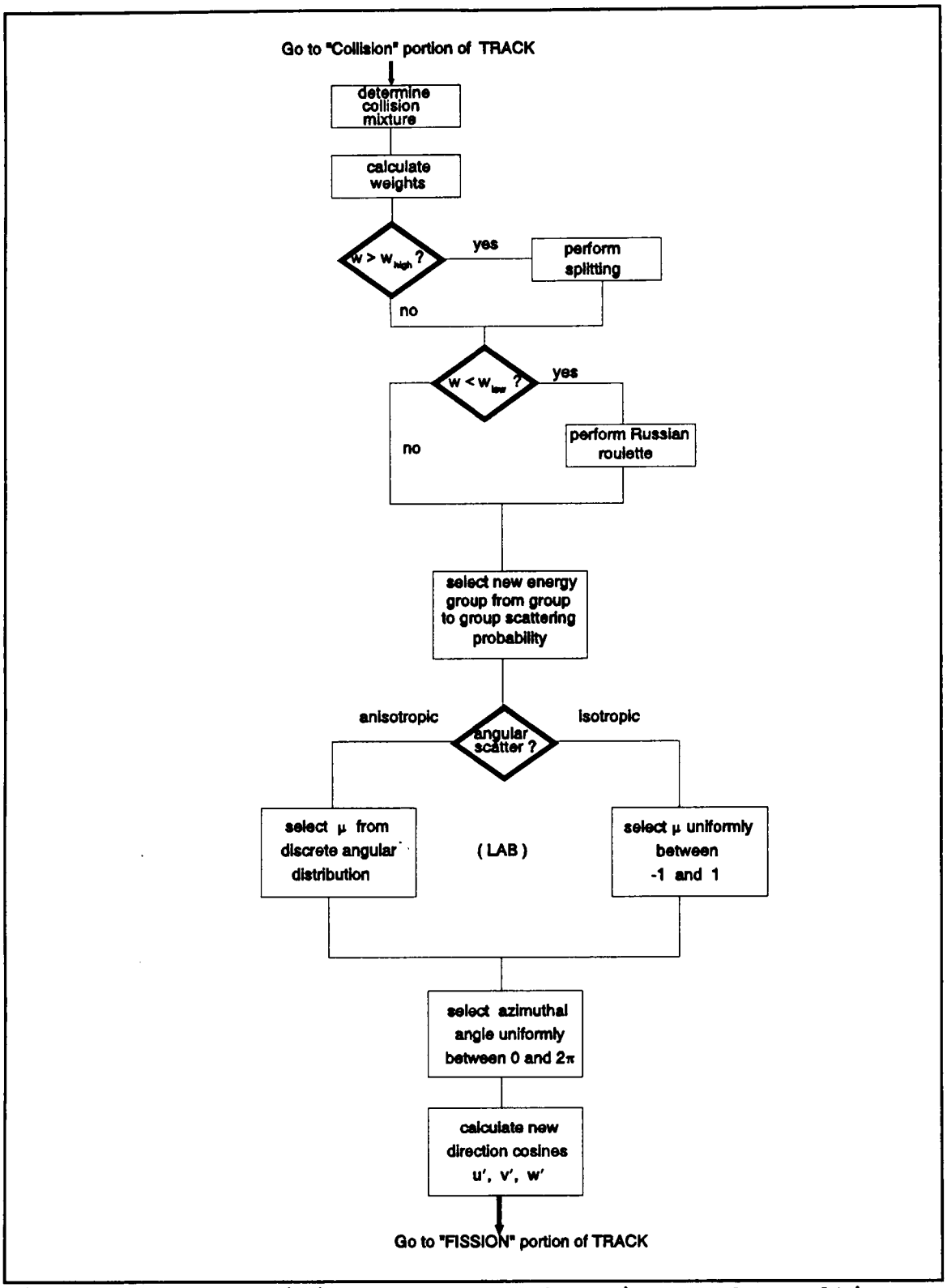


Figure 2.10: Collision Treatment Flow Diagram for Multigroup Version of KENO V.a.

group-to-group transfer. Otherwise, the direction cosine is selected uniformly on the interval from -1.0 to 1.0. Once the polar angle is sampled, the azimuthal angle is selected uniformly between 0 and 2π. Based on the azimuthal and polar angle, the new direction cosines of the particle are calculated.

During the collision treatment, the particle's fission weight is calculated and is defined as follows:

$$w_f = \frac{\nu_g \Sigma_{fg}}{\Sigma_{tg}} w_b \ . \qquad (2.3)$$

w_b = particle weight before collision.
Σ_f = macroscopic fission cross section for group g.
Σ_t = macroscopic total cross section for group g.
ν = average number of neutrons per fission for group g.

If the fission weight is less than zero, the history returns to *PATH*. A fission weight greater than zero indicates the collision occurred in fissile material, and the *FISSION* portion of TRACK is executed. During the random walk, several fission source points must be generated to provide an adequate representation of the true source distribution. A minimum production factor is defined to ensure enough fission points are generated:

$$mpf = \overline{k}\left(1 - \frac{3}{\sqrt{FG}}\right). \qquad (2.4)$$

$\overline{k}$ = running average of the calculated k_{eff}.
FG = number of histories per generation.

The minimum production factor represents the lower limit of the 99% confidence interval for the distribution of the calculated k_{eff}'s by generation.[13] The following steps describe the fission treatment in TRACK:

1. A pseudo fission weight is calculated:

$$pfw = \frac{w_f}{R} = \frac{\nu_g \Sigma_{fg} w_b}{R \Sigma_{tg}} . \qquad (2.5)$$

R = random number between 0 and 1.

2. If pfw < mpf and the history did not change supergroups, go to *PATH*.

3. If pfw < mpf and the history changed supergroups, store particle in neutron bank and start new history.

4. If pfw > mpf, redefine the pseudo fission weight:

$$pfw = \frac{mpf}{R} = \frac{\overline{k}}{R}\left(1 - \frac{3}{\sqrt{FG}}\right). \qquad (2.6)$$

5. If the fission bank (**fsbank**) is not full, determine the fission energy group from the fission spectrum, store the source point and pfw in **fsbank**, and increment the number of fission points. Go to step 8.

6. If the fission bank is full, search **fsbank** for the smallest pfw.

7. If the pfw of the new fission point is larger than the pfw from step 6, replace the old point with the new point.

8. Decrement the fission weight by mpf.

9. If the fission weight > 0, go to step 1. Otherwise, return to *PATH*.

The *ARRAY* section of TRACK is used to process a history that has exited a unit. Initially, a check is made to determine if the exited unit is part of an array or a hole. If the unit is a hole, the coordinates are updated to the current region's coordinate system and tracking returns to the *PATH* section. If the exited unit is part of an array, the location of outward crossing is determined. A check is made to determine if the particle exited the array. If the history remained in the array, the location of the particle within

the array is determined. If the new unit consists of only one region, *FINDBX* is used to determine whether the particle is crossing into another array. A particle leaving an array can either leak from the system or enter an exterior geometry region such as a reflector. If the history enters a reflector and differential albedo information is present, *ALBEDO* is used to process the reflection. Otherwise, the history proceeds to *PATH* and tracking continues.

In the *ALBEDO* section of TRACK, each boundary face of the cuboid is checked to determine whether the particle leaks from the system, experiences mirror reflection, experiences periodic reflection or undergoes differential albedo treatment. Mirror reflection results in a sign reversal of the direction cosines with no change in energy. A particle experiencing periodic reflection is moved to the opposing face of the system with no change in energy or direction. If differential albedo data is available for the reflector material, the albedo information is used to determine the returning energy and angle from the reflector as well as the new weight of the particle.

Once all particles in the neutron bank are processed, the generation is complete and control of the program is returned to GUIDE. The multiplication factor (k_{eff}) for the generation is calculated, and the next generation is processed. When the specified number of generations is complete, the results are tallied and printed in the appropriate output file.

Chapter Three

Continuous Energy KENO V.a

3.1 Input and Data Handling

In order to utilize continuous energy cross sections, extensive modifications have been made to the input processing portion of KENO V.a. In particular, several new subroutines have been developed to read and process the cross section libraries. PKENO requires mixing table input to describe the material specifications of the problem. Subroutine MIXIT which reads the mixing table in the input file expects a nuclide specified in a ZZZAAA NNC format. The ZZZAAA[†] number is a combination of atomic number and mass for the nuclide, NN identifies the data file and C denotes continuous energy.[4] The available cross section data for specific nuclides is listed in Refs. 4 and 36. There must be a blank space following the ZZZAAA number and file extension. Following the nuclide specification in the mixing table is the nuclide's atomic number density (atoms/b-cm) and temperature in degrees Kelvin. The mixing table specification for a mixture i must be in the following format:

mix=i zzzaaa nnc atom density temperature

In addition, PKENO V.a reads a thermal scattering law, S(α,β), identifier specified in a ZAID NNT format in the input file. The ZAID identifier specifies the specific compound, nn identifies the data file, and T denotes thermal scattering law data.[4]

[†]ZZZ - atomic number; AAA - atomic mass (e.g., ^{235}U = 92235).

As in the nuclide specification, there must be a blank between the ZAID identifier and file extension. The S(α,β) data is provided for a specific number of compounds, which are also listed in Refs. 4 and 36, at various temperatures. The S(α,β) data card (**sab=**) is assigned a number which corresponds to a mixture in the mixing table. The sab number identifies the mixture in which thermal scattering law data will be applied. The sab identifier must be the first data specified in the mixing table and can only apply to one mixture. If additional mixtures require the same or different scattering law data, a separate sab identifier must be supplied for each mixture. The S(α,β) specification for thermal scattering law data for mixture i must be in the following format:

sab=i zaid nnt

For example, a UO_2F_2 solution at room temperature with light water thermal scattering law treatment† would have the following mixing table input:

```
read mix
sab=1 lwtr 01t
mix=1 92235 50c 1.1760e-3 293
      92238 50c 8.2051e-5 293
       1001 50c 6.2210e-2 293
       8016 50c 3.3621e-2 293
       9019 50c 2.5161e-3 293
end mix
```

In order to use the current biasing schemes in KENO (i.e., Roulette and splitting) without modification, it is necessary to employ an energy group structure.[28] Furthermore, collecting calculated quantities (e.g., flux, leakage, etc.) by energy bins or groups is statistically more efficient. During the input processing, PKENO V.a reads a

†Light water S(α,β) treatment would be used because the Hydrogen atoms present in the mixture are bound in the water molecule.

user defined energy group structure for the continuous energy calculation.

3.2 Cross Section Processing

Based on the nuclides specified in the input, PKENO V.a reads the xsdir file† which is a sequential formatted ASCII file containing the following characteristic information for each cross section data table:[4]

1. Data table name
2. Atomic weight ratio
3. File name
4. Access route
5. File type
6. Address
7. Table length (words)
8. Record length††
9. Number of entries per record (typically 512)
10. Temperature

After reading the cross section directory file, PKENO V.a stores the information in a direct access file. Based on the xsdir information, PKENO V.a reads the cross sections, which are stored in a binary format, from the appropriate file and stores the necessary cross section data in another direct access file for easier retrieval and processing. After reading the cross sections, the fissionable nuclides are identified and the corresponding fission spectrum is generated. Then, PKENO determines the storage requirements and creates pointers for accessing the data.

---

†Cross section directory file provided with the MCNP cross sections.

††For direct access files, record length is a multiple of the number of entries per record. If there are 512 entries per record, the record length is 2048 for single-precision data on unix workstations.

Before PKENO can utilize the point cross sections, the fission spectrum must be constructed. The energy distribution from fission is governed by ENDF laws 7, 9 and 11 which correspond to a Maxwellian, Evaporation and Watt spectrum, respectively.[4,8,9] The appropriate law is specified in the cross section data for the fissionable nuclide. Each law is a function of variables which are energy dependent. These variables are tabulated in the cross section files and are used to construct the appropriate fission spectrum. After reading the cross sections, PKENO constructs the fission spectrum for each fissionable nuclide which is specified in the mixing table input. The code then normalizes the spectrum and converts it to a cumulative distribution function (cdf) which is used to select the initial energy for each source neutron in subroutine START.

3.3 Continuous Energy Random Walk

3.3.1 Source Distribution

Subroutine GUIDE performs the loop over generations and tracking of the individual neutron histories during the random walk. Prior to tracking the neutrons, GUIDE calls START which creates the initial source distribution. If a mixture in the original version of KENO contains fissile material, the mixture has an associated mixed chi spectrum. In the point version of KENO, a mixture may have several different nuclides. Consequently, each mixture is checked for the presence of fissile material on a per nuclide basis. Based on the amount of fissile material present in the system, the volume fraction of fissile material must be calculated. START checks for the presence of fissionable material in each mixture and calculates the fraction of fissile material in

the problem on a per nuclide basis. If fissile material is present in the problem, PKENO randomly selects the neutron's initial coordinates using the same spatial starting options currently in KENO V.a; however, the initial energy selection is made from the continuous energy fission spectrum, $\chi(E)$, of the fissionable nuclide present in the mixture. If more than one fissionable nuclide is present, the following relation is used to select the ith nuclide to sample:[28]

$$\sum_{j=1}^{i-1} \Sigma_f^j < R \sum_{j=1}^{N} \Sigma_f^j \leq \sum_{j=1}^{i} \Sigma_f^j, \qquad (3.1)$$

Σ_f^j = macroscopic fission cross section for j^{th} nuclide.
R = random number between zero and one.
N = total number of fissionable nuclides in mixture.

Using the above relation requires the knowledge of Σ_f^j at a particular energy. Consequently, an initial energy is assumed (i.e., 0.025 eV), and the appropriate fission cross section is read from the cross section files. After the ith nuclide is selected, the corresponding $\chi_i(E)$ is randomly sampled for the actual initial energy.

3.3.2 Collision Site Selection

Once the initial source distribution is established, GUIDE performs the loop over generations and calls subroutine TRACK which performs the tracking of the individual neutron histories within each generation. Extensive modifications were needed in TRACK to facilitate the use of point cross sections. A complete listing of the new version of TRACK is provided in Appendix A. The selection of the next collision site

is governed by the following probability density function (pdf):

$$f(x) = \Sigma_t^m(E)\, e^{-\Sigma_t^m(E)\,x}\, dx, \qquad (3.2)$$

Σ_t^m(E) = macroscopic total cross section for a mixture at energy E.
x = spatial variable.

The above pdf describes the probability that a neutron will have an interaction between x and x+dx along its flight path. Integrating Eq. (3.2) over the spatial variable yields the following cdf:

$$F(x) = \int_0^x \Sigma_t^m(E)\, e^{-\Sigma_t^m(E)\,x'}\, dx' = 1 - e^{-\Sigma_t^m(E)\,x}. \qquad (3.3)$$

The next collision site is determined by setting the cdf equal to a random number on [0,1] and solving for x. Currently, KENO V.a calculates the next collision site using the mixed macroscopic total cross section for a particular energy group. In the new version of KENO V.a, a new subroutine named MIXST is called from TRACK to calculated the mixed total cross section, Σ_t^m(E), at a particular energy. Σ_t^m(E) is used to select the next collision site. The total cross section for each nuclide in the mixture is read from the appropriate cross section file if the thermal scattering law data is not present. If $S(\alpha,\beta)$ data is present and the incident energy is below the threshold for thermal scattering treatment, the total cross section is the sum of the absorption cross section from the standard file and the elastic and inelastic cross sections from the $S(\alpha,\beta)$ file.

3.3.3 Collision Treatment

Once the collision site is determined, the collision is modeled and the post collision parameters are calculated. This portion of the development required significant modifications to subroutine TRACK. Consequently, a separate subroutine named COLLISION was developed to process the collision, and the complete FORTRAN listing of COLLISION is presented in Appendix B. Since the multigroup treatment uses mixed cross sections, a neutron interacts with the entire mixture in a collision; however, in the continuous energy approach, the collision is modeled with an individual nuclide. As a result, PKENO V.a randomly selects the collision nuclide at each collision site. If there are N different nuclides present in the mixture, the following relation is used to select the ith nuclide for interaction:[28]

$$\sum_{j=1}^{i-1} \Sigma_t^j < R \sum_{j=1}^{N} \Sigma_t^j \leq \sum_{j=1}^{i} \Sigma_t^j. \qquad (3.4)$$

Σ_t^j = Macroscopic total cross section for nuclide j.

After selecting the interacting nuclide, the neutron's weight is reduced by the nonabsorption probability:

$$w = \frac{\sigma_s^i(E)}{\sigma_t^i(E)} w_b. \qquad (3.5)$$

$\sigma_s^i(E)$ = microscopic scattering cross section for nuclide i at energy E.
$\sigma_t^i(E)$ = microscopic total cross section for nuclide i at energy E.
w_b = weight before collision.

In addition, the absorption and fission weights are calculated using Eqs. (6) and (7), respectively.

$$w_a = \frac{\sigma_a^i(E)}{\sigma_t^i(E)} w_b. \qquad (3.6)$$

σ_a^i(E) = microscopic absorption cross section for nuclide i at energy E.

$$w_f = \frac{\nu^i(E)\,\sigma_f^i(E)}{\sigma_t^i(E)} w_b. \qquad (3.7)$$

ν^i(E) = number of neutrons released per fission at energy E.
σ_f^i(E) = microscopic fission cross section for nuclide i at energy E.

Depending on the particle's weight, splitting and/or Roulette† is performed as necessary. Once the appropriate weights are calculated, the collision is processed, and the exiting energy and angle of the neutron is calculated.

In order to understand the modifications that must be made to KENO V.a, the fundamental differences between the multigroup and continuous energy collision treatment must be understood. As discussed in Chapter Two, the angular scattering in KENO V.a is treated by discretizing a P_N Legendre polynomial expansion of the continuous angular variable such that the N+1 moments of the discrete distribution are the same as the truncated polynomial. The discretization yields a set of N+1 equations giving (N+1)/2 discrete polar angles (i.e., angle cosines which are in the lab system) with corresponding (N+1)/2 probabilities.[25] The P_0 term corresponds to the group-to-group energy transfer probability. For a given group-to-group transfer, a new discrete

†Since an associated energy group structure will be used in the calculation, modifications to the current biasing schemes are not necessary.

scattering angle is randomly selected based on the associated probabilities. If the scattering is isotropic, the new direction cosine is not selected from a discrete distribution. In particular, the P_0 term determines the appropriate group-to-group transfer for isotropic scattering, and the new direction cosines are selected from an isotropic distribution instead of a discrete distribution.[13,14] At each collision site, this selection process yields the energy group and angle of the exiting neutron. However, it should be noted that nonelastic collisions† are not explicitly modeled in the KENO V.a collision treatment. Furthermore, a distinction between a nonelastic and elastic reaction is not made in the multigroup collision treatment. Nonelastic and elastic collisions are treated during the generation and processing of a group cross section library.

The collision treatment in a multigroup approach is different from the treatment required by continuous energy cross sections. With regard to MCNP cross sections, angular distribution tables are provided in the point cross section library for the target nuclide. The angular distribution tables consist of 32 equiprobable cosine bins which are provided for different incident neutron energies.[4] The angular cosines are provided in the center of mass (COM) or lab system depending on the type of reaction. For elastic scattering and discrete inelastic level scattering,†† the angular distribution is provided in the COM system.[4] Consequently, these scattering cosines must be converted to the

---

†Nonelastic collisions include inelastic scattering reactions and neutron producing reactions such as (n,2n), (n,3n), etc.

††An discrete inelastic scattering reaction is one in which the neutron excites the nucleus to a particular excitation level (e.g., first excited state, second, etc.) and is subsequently emitted.[4,8,9]

lab system using two body kinematics. By virtue of the two body kinematics equations, the exiting particle energy and direction are determined in the lab system. For other nonelastic reactions (e.g., (n,2n), (n,3n), etc.), the cross sections provide the angular distribution in the lab system which can be used directly in the Monte Carlo random walk. For these nonelastic reactions, the exiting particle energy is governed by various ENDF scattering and emission laws depending on the particular nonelastic collision. The secondary energy distribution based on these ENDF laws are provided in the MCNP cross section files.[4]

Since an explicit collision treatment is dictated by the point cross sections, elastic and nonelastic collisions are modeled explicitly in the point version of KENO V.a. To better understand the required modifications, the flow diagram for the KENO V.a collision treatment, which is presented in Chapter Two, is presented again in Figure 3.1 for clarity. In addition, the new collision treatment for the continuous energy version of KENO V.a is presented in Figure 3.2 for comparison with Figure 3.1.

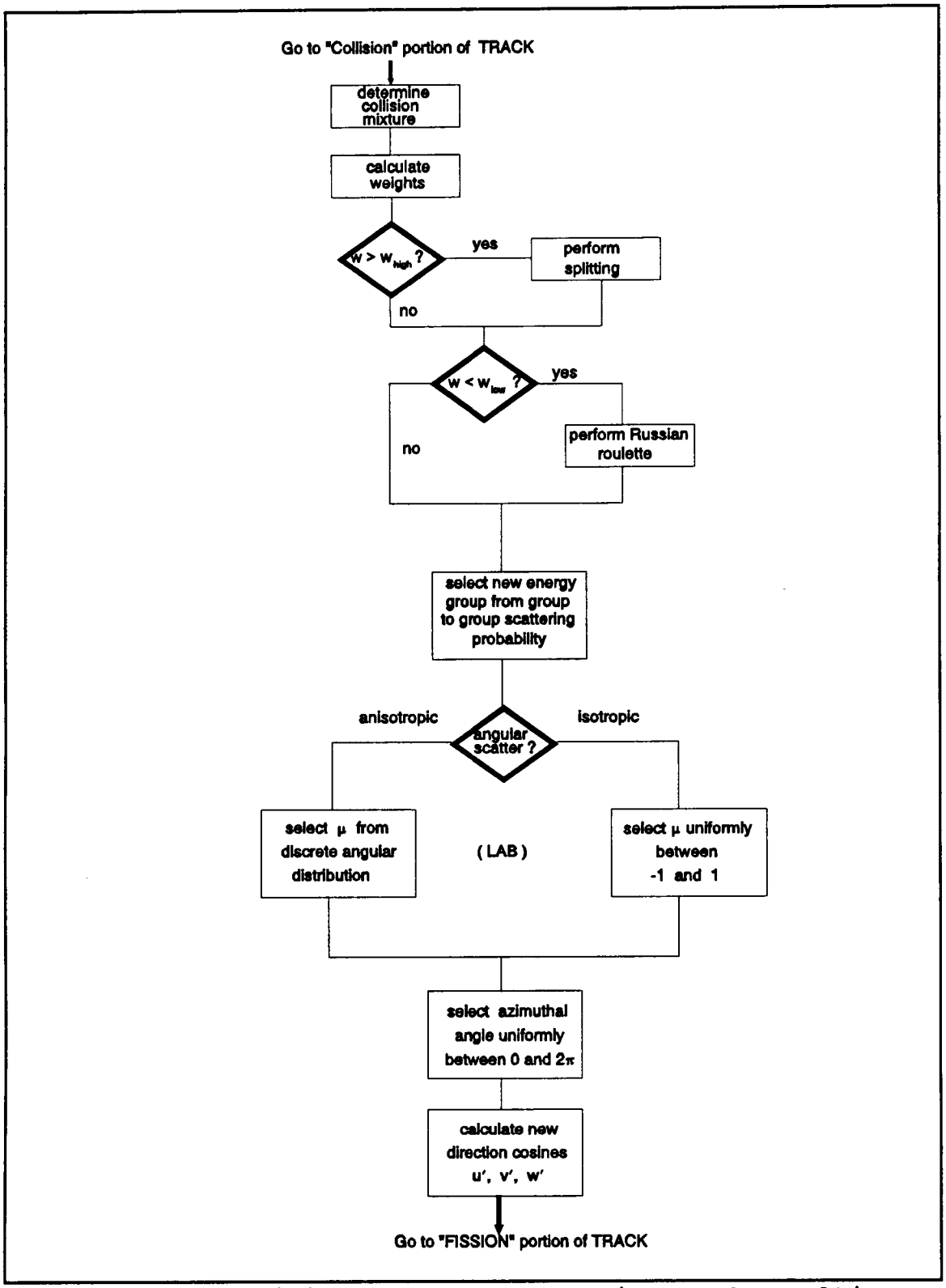


Figure 3.1: Collision Treatment Flow Diagram for Multigroup Version of KENO V.a.

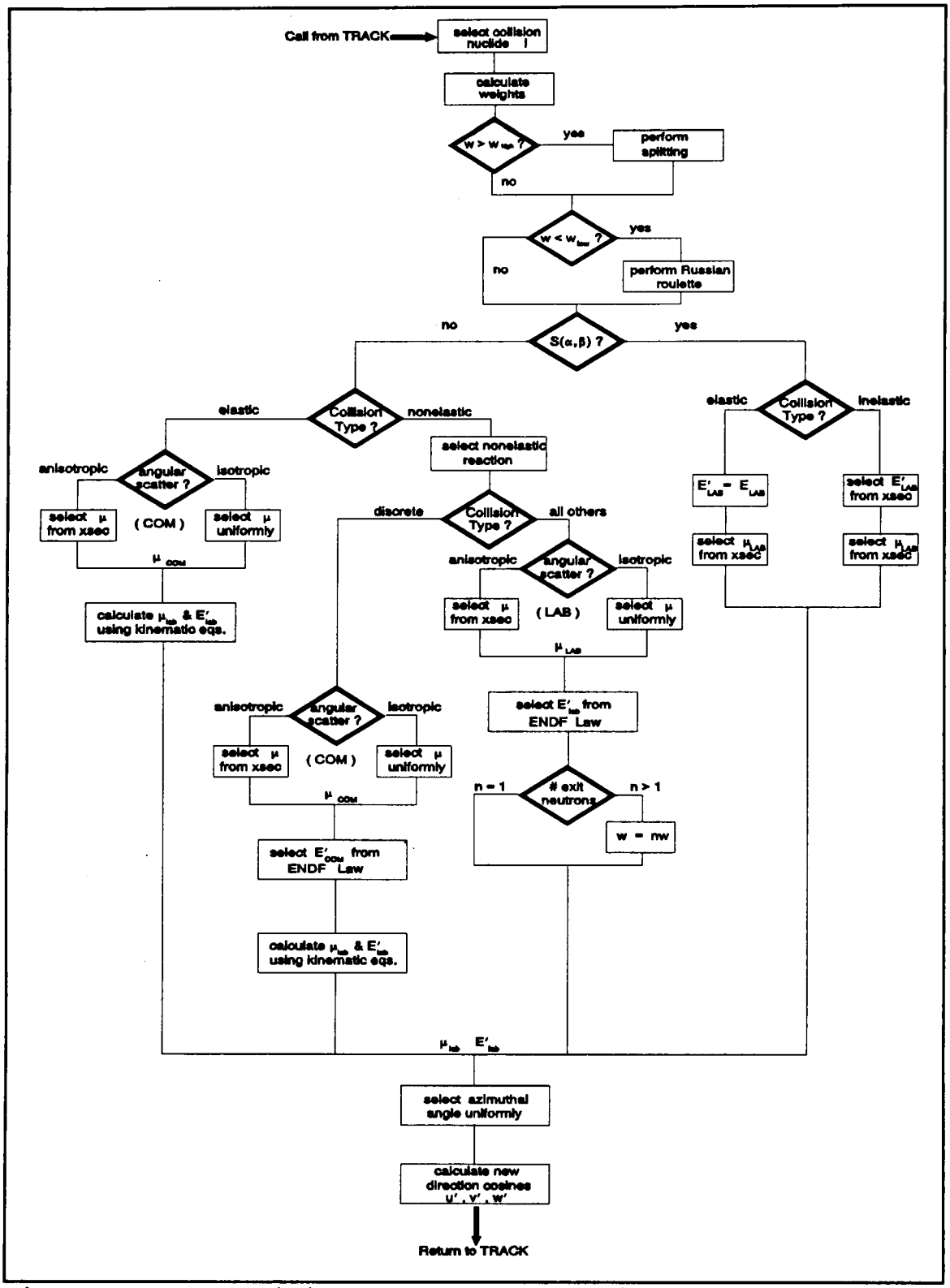


Figure 3.2: Collision Treatment Flow Diagram for Continuous Energy Version of KENO V.a.

A collision between a neutron and nucleus can be affected by the thermal motion of the target nucleus. As shown in Figure 3.2, thermal scattering law data is used to treat the thermal effects of scattering; however, the MCNP cross sections only provide $S(\alpha,\beta)$ data for a select number of nuclides in different compounds (see Refs. 4 and 36). If $S(\alpha,\beta)$ data is not available for a particular nuclide and the neutron energy is below 400 kT (i.e., k = Boltzmann constant, T = temperature in degrees Kelvin), the MCNP cross sections utilize a free gas approximation to account for the thermal motion of the target nuclide. The MCNP cross section data is provided at various temperatures (Refs. 4 and 36). Depending on the system temperature, the appropriate cross section data should be selected to correspond with the temperature of the system. The following description of sampling the speed of the target nucleus is based on the information presented in Ref. 4.

The free gas thermal treatment only applies to elastic scattering. the effective scattering cross section in the lab system for a neutron of kinetic energy E is given by

$$\sigma_s^{eff} = \frac{1}{v_n} \int_{-1}^{1} \int_{0}^{\infty} \sigma_s(v_{rel})\, v_{rel} p(V)\, dV\, \frac{d\mu_t}{2}. \qquad (3.8)$$

v_n = speed of incident neutron.
V = speed of target nucleus.
v_{rel} = relative speed between neutron and target nucleus.
μ_t = cosine of angle between neutron and target nucleus.
$\sigma_s(v_{rel})$ = scattering cross section at the relative speed.
$p(V)$ = probability density function for target speeds.

The probability density function for target speeds, p(V), is assumed to be a Maxwellian distribution given by

$$p(V) = \frac{4}{\sqrt{\pi}} \beta^3 V^2 e^{-\beta^2 V^2} \qquad (3.9)$$

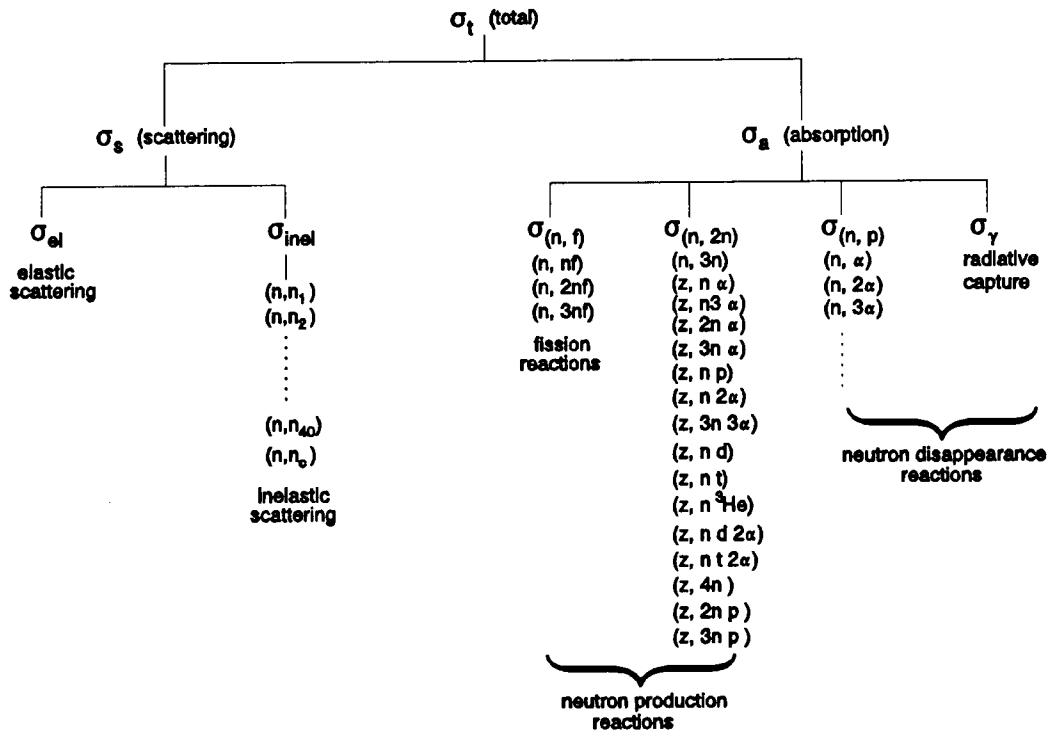
where

$$\beta = \left(\frac{AM_n}{2kT} \right)^{\frac{1}{2}}. \qquad (3.10)$$

A = mass of target nucleus.
M_n = mass of neutron.
k = Boltzmann Constant.
T = temperature of target nuclide.

The speed of the target nucleus is sampled using p(V), and the cosine of the angle between the neutron and nucleus is sampled uniformly between -1 and 1. The relative speed of the neutron is obtained by subtracting the target speed from the incident neutron speed. The elastic collision is modeled using the relative speed, and the outgoing speed of the neutron is transformed back to the lab system by adding the target speed.

The following discussion is a step-by-step procedure describing the collision treatment in Figure 3.2. Prior to describing the collision treatment, it is helpful to review the cross section hierarchy[7] which is presented in Figure 3.3.



$$\left(\begin{array}{c}\text{nonelastic}\\ \text{cross section}\end{array}\right)^{\dagger}: \sigma_{ne} = \sigma_{inel} + \text{neutron production cross sections}$$

† The nonelastic cross section is defined specifically for this work

Figure 3.3: Cross Section Hierarchy for Particle Interactions.

Collision Treatment Procedure

1.) Select the ith nuclide for collision using Eq. (3.4).

2.) Calculate the absorption weight, w_a, and fission weight, w_f, using Eqs. (3.6) and (3.7), respectively.

Reduce neutron's weight, w, by nonabsorption probability using Eq.(3.5).

3.) If $w > w_{high}$, perform splitting.†

If $w < w_{low}$, play Roulette.††

4.) If $S(\alpha,\beta)$ data is present for the nuclide and the incident energy is below the threshold for thermal scattering treatment, go to step 11.

5.) Determine the collision type (i.e., elastic or nonelastic†††). The probabilities for elastic and nonelastic collisions are given by Eqs. (3.11) and (3.12), respectively:

$$P_{el} = \frac{\sigma_{el}(E)}{\sigma_{el}(E) + \sigma_{ne}(E)} . \qquad (3.11)$$

$\sigma_{el}(E)$ = elastic scattering cross section.
$\sigma_{ne}(E)$ = nonelastic scattering cross section.

---

†Splitting is performed if the neutron bank can hold another history. If the bank is full, the neutron is not split, and a message is written. Then, a test for Roulette is performed.

††If the neutron is killed, start a new history. If neutron survives set $w = w_{avg}$. If Roulette is not played, the weight remains unchanged.

†††Refer to figure 3.3 for list of nonelastic reactions.

$$P_{ne} = \frac{\sigma_{ne}(E)}{\sigma_{el}(E) + \sigma_{ne}(E)}. \qquad (3.12)$$

If the collision is nonelastic go to step 7. If the energy is below 400 kT or the target nuclide is Hydrogen, go to step 6; otherwise, go to step 8.

6.) Sample the speed of the target nucleus and determine the relative speed between the nucleus and incident neutron. Sample the cosine of the angle between the neutron and nucleus uniformly from -1 to 1. Based on the relative speed, calculate the relative energy which will be used for elastic scattering. Go to step 8.

7.) Randomly select the kth nonelastic reaction to model using the following relation:

$$\sum_{j=1}^{k-1} \sigma_{ne}^{j} < R \sum_{j=1}^{NI} \sigma_{ne}^{j} \leq \sum_{j=1}^{k} \sigma_{ne}^{j}. \qquad (3.13)$$

σ_{nc}^{j} = j^{th} nonelastic reaction.
NI = total number of nonelastic reactions.
R = random number on [0,1].

If more than one neutron is emitted (e.g, (n,2n), (n,3n), etc.), recalculate the weight of the neutron to account for the additional neutrons (e.g., w = 2w for (n,2n) reaction, w = 3w for (n,3n)).

It should be noted that the partial fission cross sections† are nonelastic reactions. If one of the partial fission reactions is selected, the fission reaction is not

---

†Partial fission reactions: $\sigma_{n,f}$, $\sigma_{n,nf}$, $\sigma_{n,2nf}$, and $\sigma_{n,3nf}$.
Note: $\sigma_f = \sigma_{n,f} + \sigma_{n,nf} + \sigma_{n,2nf} + \sigma_{n,3nf}$.

permitted in the collision treatment because fission is explicitly treated in subroutine TRACK.

8.) For both nonelastic and elastic collisions, the procedure for selecting the exiting direction cosine is the same. If the collision is anisotropic go to step (i), otherwise go to step (ii).

i.) Anisotropic Emission

An angular distribution table is provided for each reaction. A table of 32 equiprobable cosine bins is provided for each incident energy defined on the incident energy grid for the specified reaction.[4] In other words, angular distribution tables are provided at each of the following incident energies for a particular reaction:[†]

$$E_1, E_2, \ldots\ldots, E_n, E_{n+1}, \ldots\ldots, E_{NE}$$

NE = total number of incident energies.

If the incident energy, E, is between E_n and E_{n+1}, the nth table is selected with probability P_n, and the n+1 table is selected with probability P_{n+1}:

$$P_n = \frac{E_{n+1} - E}{E_{n+1} - E_n}. \qquad (3.14)$$

$$P_{n+1} = \frac{E - E_n}{E_{n+1} - E_n}. \qquad (3.15)$$

[†]Incident energies are specified in the lab system.

Once the angular distribution table is selected, the ith cosine bin is randomly selected (i.e., i = 32R + 1).

The cosine of the exiting angle is determined using the following equation:

$$\mu = \mu_i + R(\mu_{i+1} - \mu_i) . \qquad (3.16)$$

μ_i = cosine of exiting angle for the ith bin.
μ_{i+1} = cosine of exiting angle for the i+1 bin.

i.) Isotropic Emission

Select μ uniformly between -1 and 1 (i.e., μ = 2R - 1).

9.) Read flag in cross section file which identifies the system for the exiting angle:

COM

Elastic scattering

Discrete inelastic scattering from levels

LAB

All other nonelastic collisions

10.) Process the collision:

i.) Elastic Scattering ($\mu = \mu_{com}$)

The following two equations are used to determine the exiting angle and energy for elastic scattering, respectively:[4,29]

$$\mu_{lab} = \frac{1 + A\mu_{com}}{(1 + A^2 + 2A\mu_{com})^{1/2}} . \qquad (3.17)$$

A = atomic mass of nuclide.
μ_{lab} = cosine of exiting angle in lab system.
μ_{com} = cosine of exiting angle in com system.

$$E'_{lab} = \frac{1}{2} E_{lab} [(1 - \alpha)\mu_{com} + \alpha] . \qquad (3.18)$$

$$\alpha = \left(\frac{A-1}{A+1}\right)^2 .$$

E'_{lab} = exiting energy in lab system.
E_{lab} = incident energy in lab system.

If the speed of the target nucleus was not sampled go to step 12; otherwise, add speed of nucleus to outgoing speed of neutron to account for thermal motion of target nucleus. Based on new outgoing speed, calculate the exiting energy of the neutron. Go to step 12.

ii.) Discrete Inelastic Scattering ($\mu = \mu_{com}$)

For discrete inelastic scattering, the exiting energy is sampled in the COM system using ENDF law 3:[4,29]

$$E'_{com} = \left(\frac{A}{A+1}\right)^2 E_{lab} - \left(\frac{A}{A+1}\right)^2 Q \qquad (3.19)$$

Q = value for the inelastic reaction.

The exiting energy and angle are converted to the lab system using the following kinematic equations, respectively:[4,7,30]

$$E'_{lab} = E'_{com} + \left[\frac{E_{lab} + 2\mu_{com}\sqrt{E_{lab}E'_{com}}}{(A+1)^2}\right], \qquad (3.20)$$

$$\mu_{lab} = \mu_{com}\sqrt{\frac{E'_{com}}{E'_{lab}}} + \frac{1}{A+1}\sqrt{\frac{E_{lab}}{E'_{lab}}} . \qquad (3.21)$$

Go to step 12.

iii.) All other Nonelastic Collisions ($\mu = \mu_{lab}$)

E'_{lab} is sampled form the appropriate ENDF law.

Go to step 12.

11.) Read the elastic and inelastic cross sections at incident energy E from the $S(\alpha,\beta)$ data file. Determine if the collision is elastic or inelastic using the probabilities which are given by Eqs. (3.11) and (3.12), respectively.

i.) Elastic

No change in outgoing energy is assumed (i.e. $E'_{lab} = E_{lab}$)

Based on the incident energy, sample the outgoing angular cosine, μ_{lab}.

ii.) Inelastic

Based on the incident energy, sample the outgoing energy, E'_{lab}

Sample the cosine of the exiting angle from the equally probable discrete cosines tabulated with E'_{lab}.

12.) Select the azimuthal angle η uniformly between 0 and 2π (i.e., $\eta = 2R\pi$)

13.) Calculate the new direction cosines:[13,14]

$$u' = u\mu - \sqrt{v^2 + w^2}\sqrt{1 - \mu^2}\cos\eta. \qquad (3.22)$$

$$v' = v\mu + \frac{uv}{\sqrt{v^2 + w^2}}\cos\eta\sqrt{1 - u^2} - \frac{w}{\sqrt{v^2 + w^2}}\sqrt{1 - u^2}\sin\eta. \qquad (3.23)$$

$$w' = w\mu + \frac{uw}{\sqrt{v^2 + w^2}}\cos\eta\sqrt{1 - \mu^2} + \frac{v}{\sqrt{v^2 + w^2}}\sqrt{1 - \mu^2}\sin\eta. \qquad (3.24)$$

u, v, and w are the initial direction cosines.
u', v', and w' are the exiting direction cosines.
$\mu = \mu_{lab}$

3.4 Program Flow

As stated in Section 2.3, the multigroup version of KENO V.a consists of 175 FORTRAN 77 subroutines. In the development of PKENO V.a, numerous modifications have been made to the original KENO subroutines, and 25 additional subroutines have been developed to facilitate the use of point cross sections. The program flow of KENO, which is presented in Chapter Two, is maintained as much as possible in PKENO V.a. The following description is a brief overview of the new subroutines and their location within the code.

As in KENO, the first portion of PKENO V.a initializes the program prior to calling MASTER. The flow chart for program initialization of PKENO follows the same logic which is presented in Figure 2.6. MASTER is called to coordinate the logical program flow of PKENO, and an abbreviated diagram of MASTER is presented in Figure 3.4. A brief description of each subroutine is summarized in Table 3.1 with the new subroutines highlighted in boldface type. The reading of the energy group structure and calculation of inverse speeds is performed in subroutine INVEL. Based on the mixing table input, XSPROC is called to read and process the cross section data. If restart data is available, RDRSTI is called to read the appropriate restart information. Subroutines WRTRSTI and WRTRSTP are used to store problem dependent data in a file if restart information is to be saved in a restart data file.

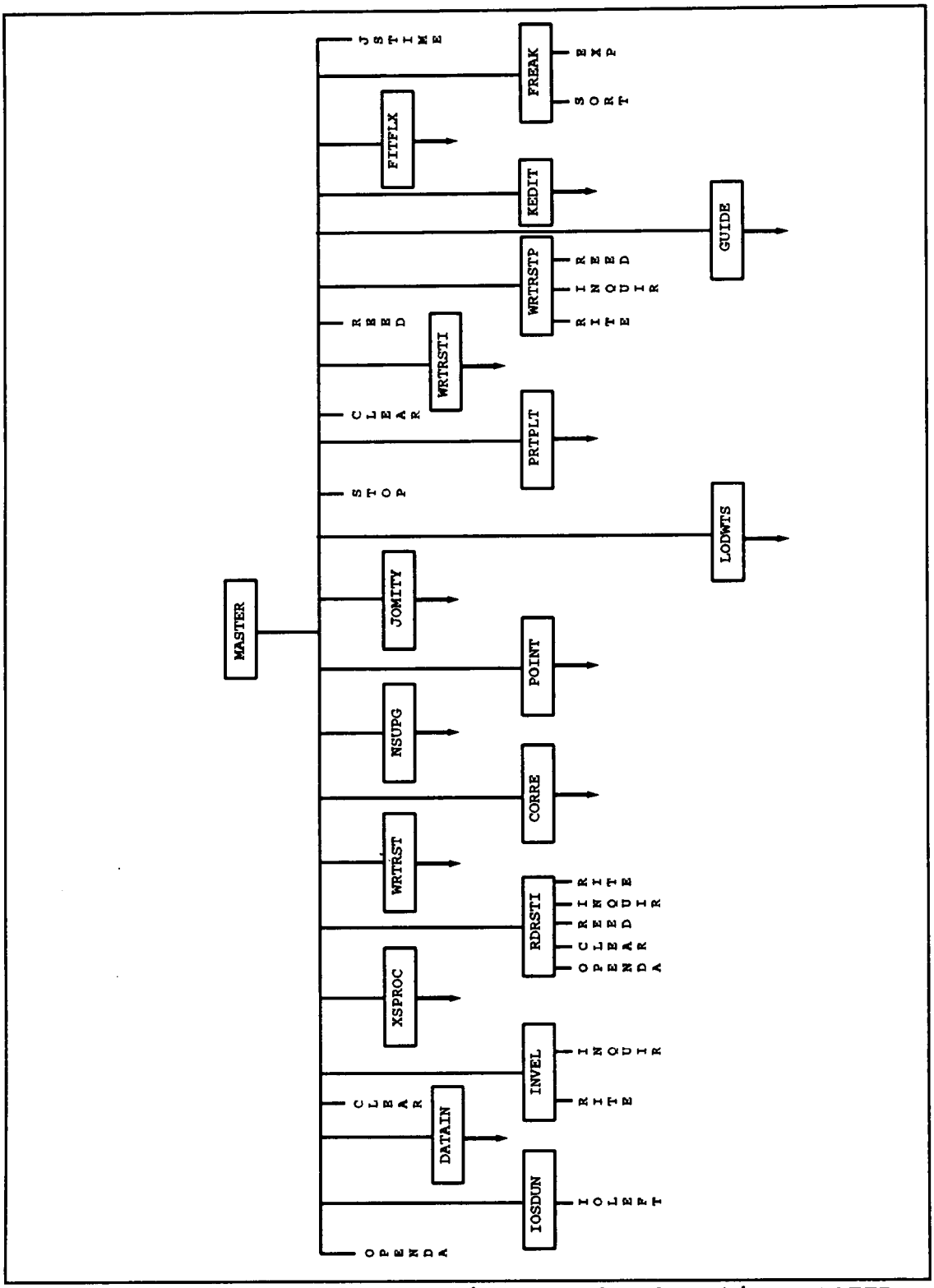


Figure 3.4: PKENO V.a Flow Diagram of Subroutine MASTER.

Table 3.1 Description of Subroutines Called by MASTER in PKENO V.a

Subroutine	Description
OPENDA	open direct access files
IOSDUN	initialize I/O's
DATAIN	read input data
INVEL	calculate inverse speeds
XSPROC	read and process cross sections
RDRSTI	read information for restart problem
WRTRST	write restart data
CORRE	generate albedo data
NSUPG	generate supergroup information
POINT	calculate pointers
JOMITY	process geometry information
PRTPLT	print user defined 2-D plots
CLEAR	initialize arrays for calculated data
LODWTS	load biasing data
WRTRSTI	initialize direct access for writing restart information
WRTRSTP	write data for restart problem
GUIDE	control tracking
KEDIT	edit calculated data
FITFLX	load fluxes for printing
FREAK	generate k_{eff} frequency distribution
JSTIME	determine time used by code

As in KENO V.a, DATAIN controls the reading of the input data except for the title and parameter information. The only addition to DATAIN is RDXSDIR which searches the cross section directory file, xsdir, for information about the cross section data specified in mixing table. A flow diagram for RDXSDIR is presented in Figure 3.5, and a summary of the associated subroutines is presented in Table 3.2. RDXSDIR calls NXTSYM to locate the next symbol in a character string and return the corresponding position. Based on the mixing table information, FILTAB, FILEXT and FSABID are used to identify and store the nuclide file, nuclide file extension, and S(α,β) identifier, respectively.

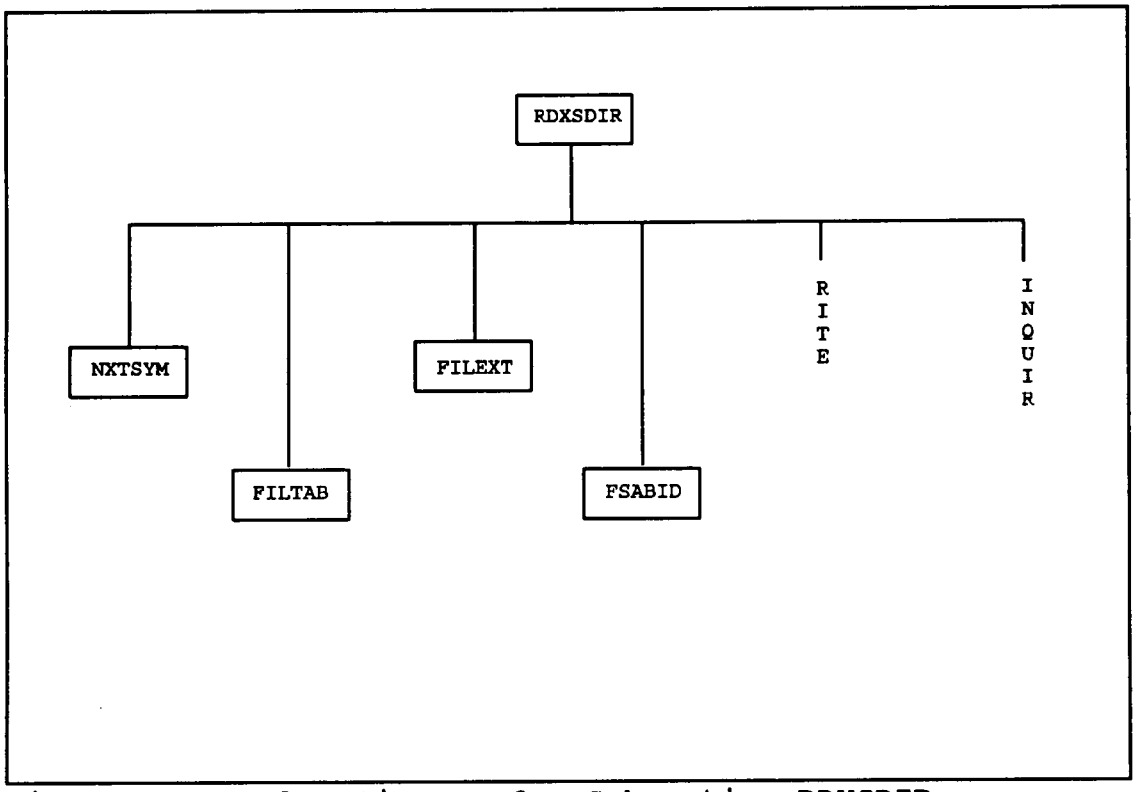


Figure 3.5: Flow Diagram for Subroutine RDXSDIR.

Table 3.2 Description of Subroutines Called by RDXSDIR.

Subroutine	Description
NXTSYM	determine location in character string
FILTAB	identify cross section file name
FILEXT	identify cross section extension
FSABID	identify S(α,β) identifier
RITE	write array in direct access file
INQUIR	determine next direct access record

After processing the input data, MASTER calls XSPROC to process the cross section information. A flow diagram for XSPROC is presented in Figure 3.6, and a description of each subroutine is presented in Table 3.3. As shown in Figure 3.6, XSREAD is called to read the cross section information. Based on the fissile material specified in the problem, XSPROC calls CHIGEN to read the appropriate ENDF Law governing fission and construct the fission spectrum for each fissionable nuclide. CHIGEN then normalizes the spectrum and converts it to a cumulative distribution function (cdf). Then, XSPROC calls XSWRITE to store the cross section data in the direct access file.

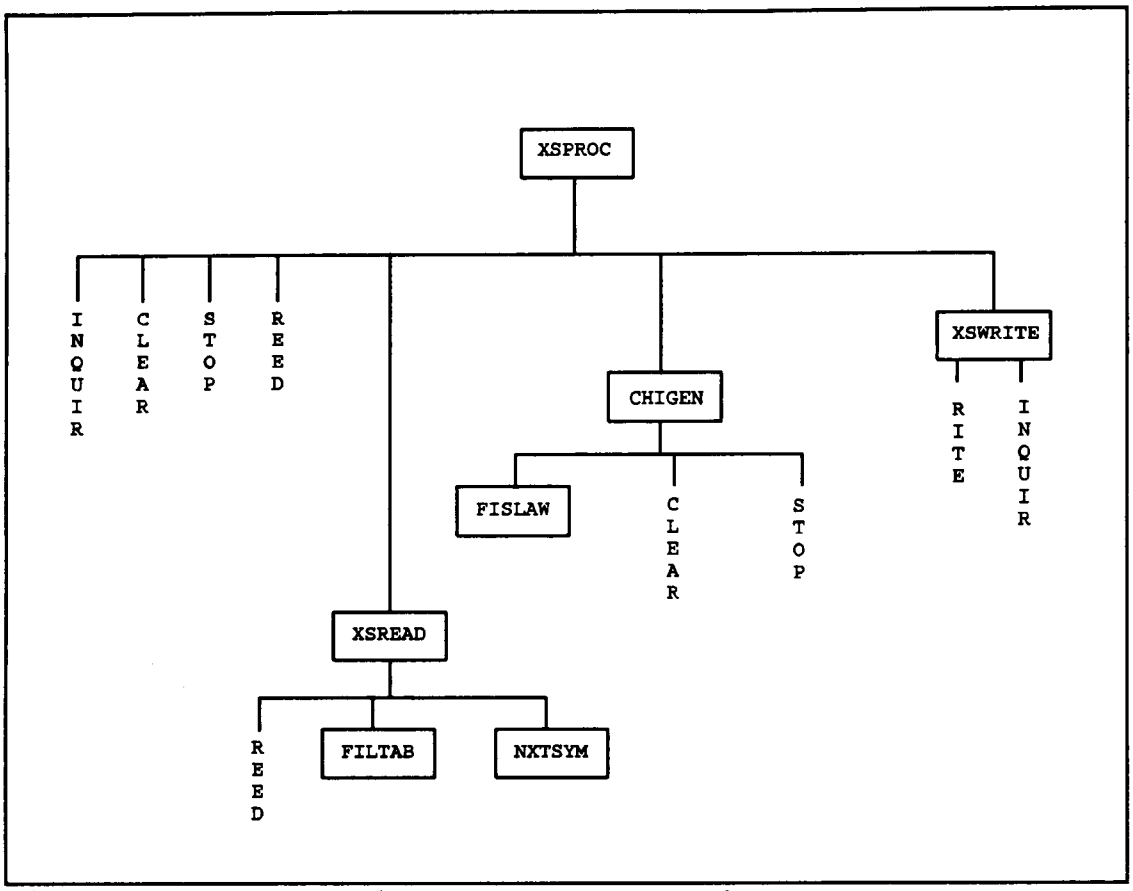


Figure 3.6: Flow Diagram for Subroutine XSPROC.

Table 3.3 Description of Subroutines Called by XSPROC.

Subroutine	Description
XSREAD	read cross section data
CHIGEN	generate chi spectrum
XSWRITE	write cross sections in direct access file
NXTSYM	determine location in character string
FILTAB	identify cross section file name
FISLAW	read fission spectrum data from cross sections

Once the cross section information is processed, control of the program returns to MASTER. PKENO then processes the geometry data, pointers, plot data, etc. prior to calling GUIDE. As in KENO, GUIDE calls TRACK to perform the tracking of each history through the system. A flow diagram for TRACK is presented in Figure 3.7, and a description of the subroutines is tabulated in Table 3.4. The flow of TRACK is the same as KENO; however, subroutine COLLISION is used to model the neutron interaction with a nuclide. When a fission occurs, SELLAW is used to determine which ENDF law governs the fission event, and ENDFLAW is used to determine the exiting energy of the fission neutron(s).

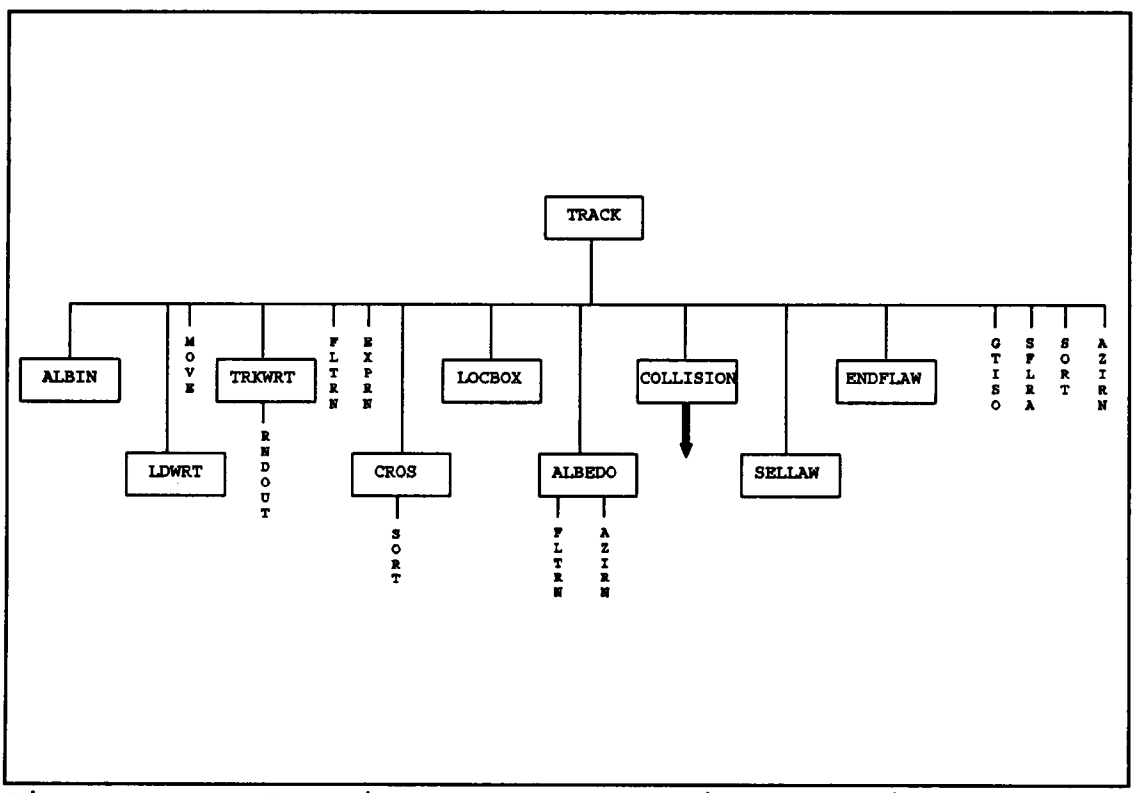


Figure 3.7: Flow Diagram for Subroutine TRACK in PKENO V.a.

Table 3.4 Description of Subroutines Called by TRACK in PKENO V.a

Subroutine	Description
ALBIN	initialize albedo information
LDWRT	print debug information
MOVE	move data in and out of storage
TRKWRT	print information about the current neutron
FLTRN	provide random number between zero and one
EXPRN	provide random number from exponential distribution
CROS	processes inward and outward geometry crossing
LOCBOX	determine the current geometrical unit
ALBEDO	process albedo reflection
COLLISION	process collision with nuclide
SELLAW	select appropriate ENDF law
ENDFLAW	process selected ENDF law
GTISO	provide isotropic direction cosines
SFLRA	provide random number between -1.0 and 1.0
SQRT	calculate direction cosines
AZIRN	provide sine and cosine of a random azimuthal angle

When a collision occurs, subroutine COLLISION is used to process the neutron interaction with a nuclide. Figure 3.8 describes the program flow of COLLISION, and Table 3.5 summarizes the function of each subroutine in Figure 3.8. For the incident energy of the neutron, LOCAT searches the energy grid of the cross section table for the location of the incident energy. Subroutine INTERP is used to perform the required interpolation according to the information provided in the cross section data. If the interpolation scheme varies as a function of energy, LOCNBT determines which scheme

should be used for the incident energy. MOVE is used throughout COLLISION to move data in and out of storage. If the neutron energy is below 400 kT (i.e., k=Boltzmann constant and T is degrees Kelvin) and $S(\alpha,\beta)$ is not available, subroutine SAMPVEL is called to sample the speed of the target nuclide. SELLAW is used to determine which ENDF law governs the collision event, and ENDFLAW is called to process the appropriate ENDF law. Subroutine ROTASZ is used to sample the direction cosines at a specified angle from an axis, and AZIRN provides the sine and cosine of a random azimuthal angle. After completing the collision treatment, control of the program is returned to TRACK where the neutron tracking continues.

Table 3.5 Description of Subroutines Called by COLLISION in PKENO V.a

Subroutine	Description
CLEAR	clear specified arrays
LOCAT	locate position in cross section array
INTERP	perform specified interpolation
LOCNBT	identify interpolation scheme to be used
MOVE	move data in and out of storage
FLTRN	provide random number between zero and one
TRKWRT	print information about the current neutron
SAMPVEL	sample target nuclide speed
STOP	stop program if error occurs
SELLAW	select appropriate ENDF law
ENDFLAW	process selected ENDF law
ROTASZ	sample a direction at a specified angle from an axis
AZIRN	provide sine and cosine of a random azimuthal angle

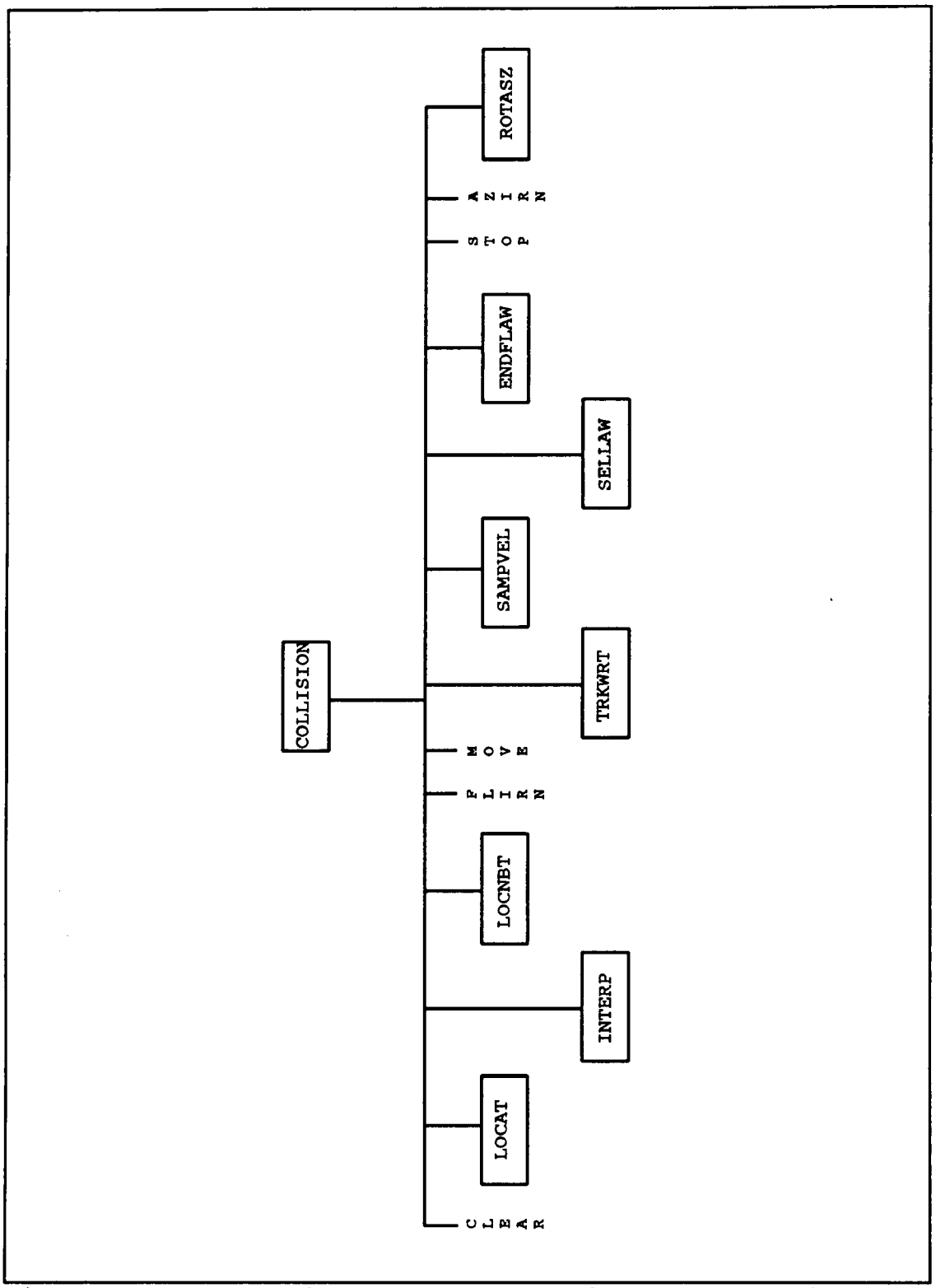


Figure 3.8: Flow Diagram for Subroutine COLLISION in PKENO V.a.

Chapter Four

Results

4.1 KENO V.a Benchmark Problems

A set of 25 problems accompanies the standard KENO V.a package and is used to benchmark the code for a particular computing platform. These test problems represent a wide variety of criticality problems.[13] Moreover, the problems test all the geometry types as well as the extended geometry options which include arrays, holes, nested holes and nested arrays. In order to test the geometry package, some of the problems are actually different geometrical representations of the same problem. A brief description of each sample problem is presented in Table 4.1. Most of the test cases are based on critical experimental data; however, a few of the problems are either variations on experimental data or subcritical systems such as problems 6 and 8. In addition, problems 9 and 16 are geometrically infinite configurations with case 9 resulting in a supercritical configuration. It should be noted that the sample input for problem 18 deviates significantly from the corresponding critical experiment.[27]

All of the calculations in this dissertation were performed on a Sun SPARC 20 model 50 workstation. In order to test the new code, PKENO V.a was used to calculate each of the 25 KENO V.a test problems. For comparison, KENO V.a and MCNP were also used to calculate the benchmark problems. The calculated k_{eff}'s for the test problems are presented in Table 4.2. All of the problems were modeled using 30,000 histories unless noted otherwise. In addition, PKENO and MCNP used continuous

energy ENDF/B-V cross sections, and KENO V.a used the 27 group ENDF/B-IV cross section library.

Each of the 25 input files for PKENO V.a, KENO V.a and MCNP are presented in Appendices C, D and E, respectively. The calculated k_{eff}'s obtained with PKENO V.a are in agreement (i.e., within $\pm 2\sigma$) with the results calculated with KENO V.a and MCNP except for problem 14. The calculated results obtained with PKENO V.a and KENO V.a for problem 14 are in agreement; however, the MCNP multiplication factor is $\sim 2\%$ low. Of the 25 test problems, five are not critical experiments or deviate from critical experiment data (i.e., cases 6, 8, 9, 16 and 18). Based on the remaining twenty problems, PKENO V.a has a 0.24% average deviation from critical (i.e., $k_{eff} = 1.0$). In comparison, KENO V.a and MCNP have a 0.36 and 0.10% average deviation from critical, respectively. These results demonstrate the accuracy of PKENO for calculating the KENO V.a series of test problems.

Table 4.1: Description of the 25 KENO V.a Benchmark Problems.[13]

Problem	Problem Description
1	Bare 2x2x2 array of uranium metal cylinders
2	Same as problem 1 with different geometrical description and options
3	2x2x2 array of uranium cylinders with 15.24 cm paraffin reflector
4	Same as problem 3 except use different geometry description
5	Same as problem 3 & 4 with 30.48 cm paraffin reflector
6	Single unit from problem 1
7	Single unit from problem 1 mirror reflected on x, y and z faces
8	Infinitely long metal cylinder from problem 1
9	2x2x2 array of uranium cylinders specularly reflected on all faces
10	Same as problem 1 with restart information written every 5_{th} generation
11	Restart problem 10 with 51st generation
12	Array of four uranium metal cylinders and four cylinders of $UO_2(NO_3)_2$ solution
13	Two uranium metal cuboids in uranium metal annulus
14	Uranium metal cylinder in uranium metal annulus
15	Water reflected uranium metal sphere supported by plexiglass collar
16	Infinite number of slabs of $U(93.2)O_2F_2$ solution
17	Sphere of $U(93.2)O_2F_2$ solution
18	3x3x3 array of cylinders of $UO_2(NO_3)_2$ solution reflected by paraffin
19	Same as problem 12 using array of arrays option
20	Triangular pitched array of 7 $U(93.2)O_2F_2$ cylinders
21	Unreflected sphere of $U(4.89)O_2F_2$ solution
22	Same as problem 1 with different geometrical description
23	Same as problem 1 with different geometrical description
24	Same as problem 1 with different geometrical description
25	Same as problem 1 with different geometrical description

Table 4.2: Calculated k_{eff}'s for the 25 KENO V.a Benchmark Problems.

Problem	PKENO V.a	KENO V.a	MCNP
1	0.9954 ± 0.0039	1.0001 ± 0.0048	0.9985 ± 0.0036
2	0.9954 ± 0.0039	1.0001 ± 0.0048	0.9985 ± 0.0036
3	0.9988 ± 0.0049	1.0037 ± 0.0052	1.0044 ± 0.0047
4	1.0020 ± 0.0051	1.0121 ± 0.0055	1.0077 ± 0.0120
5	0.9989 ± 0.0047	1.0053 ± 0.0052	1.0077 ± 0.0120
6	0.7396 ± 0.0045	0.7496 ± 0.0037	0.7434 ± 0.0028
7	0.9916 ± 0.0044	0.9939 ± 0.0040	0.9984 ± 0.0037
8	0.9349 ± 0.0042	0.9491 ± 0.0041	0.9444 ± 0.0038
9	2.2832 ± 0.0032	2.2970 ± 0.0041	2.2949 ± 0.0045
10	0.9954 ± 0.0039	1.0001 ± 0.0048	0.9985 ± 0.0036
11	0.9954 ± 0.0039	1.0001 ± 0.0048	0.9985 ± 0.0036
12	1.0007 ± 0.0057	0.9962 ± 0.0053	0.9971 ± 0.0056
13	0.9963 ± 0.0038	1.0024 ± 0.0041	0.9957 ± 0.0035
14	0.9997 ± 0.0043	1.0032 ± 0.0046	0.9836 ± 0.0035
15	0.9926 ± 0.0044	1.0036 ± 0.0046	1.0021 ± 0.0043
16	0.9975 ± 0.0047	0.9955 ± 0.0026	0.9935 ± 0.0044
17	1.0101 ± 0.0060	1.0164 ± 0.0053	0.9967 ± 0.0056
18	1.0180 ± 0.0052	1.0197 ± 0.0069	1.0203 ± 0.0047
19	1.0002 ± 0.0054	1.0222 ± 0.0054	0.9971 ± 0.0056
20	0.9964 ± 0.0054	1.0032 ± 0.0060	1.0040 ± 0.0056
21	0.9925 ± 0.0037	0.9950 ± 0.0035	0.9893 ± 0.0034
22	0.9958 ± 0.0041	1.0008 ± 0.0044	0.9986 ± 0.0036
23	0.9990 ± 0.0040	1.0065 ± 0.0045	0.9985 ± 0.0036
24	1.0020 ± 0.0043	1.0029 ± 0.0042	1.0014 ± 0.0034
25	0.9934 ± 0.0039	1.0046 ± 0.0046	1.0028 ± 0.0037

4.2 MCNP Benchmark Problems

The MCNP code package comes with a primer[34] which assists the nuclear criticality safety analyst with MCNP criticality calculations. Since the reliance on Monte Carlo simulation is increasing, the primer teaches by example and helps the analyst understand the basic concepts in calculating k_{eff} confidence intervals using MCNP. A variety of example problems ranging from simple cylinders to 3 dimensional arrays are presented. The primer provides a thorough description of each problem and a step-by-step procedure for generating the MCNP input file. The example problems include Pu metal, uranyl fluoride and plutonium nitrate solutions in various geometrical configurations. The atomic number densities are provided with most of the cases as well as complete descriptions of the system geometry. Consequently, the primer is an excellent source for benchmark problems.

A series of seven example problems from the primer were calculated using PKENO V.a, KENO V.a and MCNP. Detailed information for each problem is provided in Ref. 34, and a brief description of each problem is provided in Table 4.3. Each of the seven input files for PKENO V.a, KENO V.a and MCNP are presented in Appendices C, D and E, respectively. The calculated k_{eff}'s for each problem are presented in Table 4.4, and each Monte Carlo simulation used 150,000 histories unless noted otherwise.

In addition to the primer, an MCNP neutron benchmark report by Whalen, Cardon, Uhle and Hendricks[35] provides a series of test problems for the code. The report tests MCNP on criticality, pulsed sphere and neutron shielding problems. The

criticality problems presented in Ref. 35 are useful for testing PKENO V.a. The benchmark problems consist of critical experiments for the following types of fissile systems:

1. Fast neutron systems.
2. Low enrichment systems.
3. Reflected neutron systems.
4. Interacting systems.

A series of seven benchmark problems from Ref. 35 were calculated using PKENO V.a, KENO V.a and MCNP. A brief description of each problem is presented in Table 4.5, and the different input files for PKENO V.a, KENO V.a and MCNP are presented in Appendices C, D and E, respectively. 30,000 histories were used for each calculation except for problems p12, p31 and p41, which used 150,000 histories. The calculated k_{eff}'s for each benchmark problem are presented in Table 4.6.

As shown in Tables 4.4 and 4.6, the PKENO V.a calculated results for the MCNP benchmark problems agree (i.e., within $\pm 2\sigma$) with the calculated k_{eff}'s obtained with KENO and MCNP. Of the fourteen cases, problem ex2.2 is the only subcritical case. With regard to predicting criticality, PKENO has a 0.75% average deviation from a $k_{eff}=1.0$. While the MCNP and KENO average deviation from experiment is 0.94% and 0.68%, respectively. These results demonstrate the accuracy of PKENO for calculating the various MCNP benchmark problems.

Table 4.3: Description of the MCNP Primer Benchmark Problems.[34]

Problem	Problem Description
ex1	Pu metal sphere with nickel coating
ex2.1	Pu metal cylinder
ex2.2	Pu metal cylinder radially reflected by natural uranium
ex2.3	Pu metal cylinder fully reflected by natural uranium
ex3	Bare aluminum cylinder partially filled with $U(4.89)O_2F_2$ solution
ex4	2 $U(93.4)O_2F_2$ cylinders inside a tank of water
ex5	3x2 array of plutonium nitrate solution cylinders

Table 4.4: Calculated k_{eff}'s for MCNP Primer Benchmark Problems.

Problem	PKENO V.a	KENO V.a	MCNP
ex1	1.0028 ± 0.0026	0.9953 ± 0.0021	1.0041 ± 0.0016
ex2.1	1.0243 ± 0.0023	1.0119 ± 0.0020	1.0217 ± 0.0017
ex2.2	0.8905 ± 0.0019	0.8830 ± 0.0021	0.8921 ± 0.0015
ex2.3	1.0299 ± 0.0021	1.0198 ± 0.0021	1.0343 ± 0.0020
ex3†	0.9952 ± 0.0044	0.9941 ± 0.0040	1.0022 ± 0.0047
ex4†	1.0188 ± 0.0050	1.0172 ± 0.0057	1.0166 ± 0.0057
ex5	0.9965 ± 0.0025	1.0028 ± 0.0024	0.9949 ± 0.0029

†30,000 histories

Table 4.5: Description of the MCNP Neutron Benchmark Problems.[35]

Problem	Problem Description
p12	Jezebel metal sphere with 95.5% ^{239}Pu
p13	Jezebel metal sphere with 80.0% ^{239}Pu
p21	Bare uranium metal cylinder with 10.9% ^{235}U
p22	Bare uranium metal cylinder with 14.11% ^{235}U
p31	U(93.5) metal sphere reflected with graphite
p32	U(97.67) metal sphere reflected with water
p41	3 $U(93.2)O_2F_2$ cylinders arranged in triangular pitch

Table 4.6: Calculated k_{eff}'s for MCNP Neutron Benchmark Problems.

Problem	PKENO V.a	KENO V.a	MCNP
p12	1.0096 ± 0.0024	1.0043 ± 0.0020	1.0113 ± 0.0017
p13	1.0081 ± 0.0052	1.0219 ± 0.0049	1.0153 ± 0.0037
p21	1.0003 ± 0.0032	0.9927 ± 0.0030	0.9981 ± 0.0033
p22	1.0024 ± 0.0031	0.9943 ± 0.0035	1.0038 ± 0.0033
p31	0.9939 ± 0.0017	1.0041 ± 0.0018	0.9963 ± 0.0016
p32	0.9967 ± 0.0041	1.0076 ± 0.0044	1.0036 ± 0.0048
p41	1.0193 ± 0.0025	1.0229 ± 0.0026	1.0209 ± 0.0024

4.3 Additional Benchmark Problems

Twelve additional computational problems are available from the Evaluation Techniques Working Group for the Nuclear Criticality Technology and Safety Project (sponsored by the U. S. Department of Energy).[10] The problems are derived from critical experiments but should not be construed as experimental benchmarks. Instead, the problems are designed to compare different computational techniques. The benchmark cases involve uranium and plutonium in either metal or solution states. In addition, the geometrical configurations range from single units to arrays. Since a comprehensive description of each case is available, the 12 problems provided by Ref. 10 are excellent computational benchmarks for PKENO V.a.

Of the twelve problems, case ten consists of a hexagonal lattice of 301 PuO_2-UO_2 fuel rods in Pu-U-Gd nitrate solution. A hexagonal array in PKENO and KENO can only be modeled using the "hole" extended geometry option discussed in Chapter Two. As a result, 301 fuel pins are difficult to model using the PKENO/KENO geometry package because the location of each hole must be identified in the input file. In other words, a repeated hexagonal array structure does not exist for the KENO V.a geometry package. Consequently, problem ten is not modeled. In addition, the twelfth problem from Ref. 10 is presented in the previous section as case ex5. As a result, only 10 of the twelve problems from Ref. 10 are modeled using PKENO, KENO and MCNP. A brief description of each problem is presented in Table 4.7, and the different input files for PKENO V.a, KENO V.a and MCNP are presented in Appendices C, D and E, respectively. The calculated k_{eff}'s for each case are presented in Table 4.8 along with

the associated number of histories used in the Monte Carlo model. As shown in Table 4.8, the calculated k_{eff}'s obtained with PKENO V.a are in agreement (i.e., within $\pm 2\sigma$) with the results obtained with KENO and MCNP.

Table 4.7: Description of Additional Computational Benchmark Problems.[10]

Problem	Problem Description
1	Water reflected plutonium metal sphere
2	Plutonium nitrate solution in steel sphere with water reflection
3	Bare plutonium nitrate solution in aluminum sphere
4	Bare ^{233}U nitrate solution in aluminum sphere
5	Bare ^{235}U metal sphere (godiva)
6	Bare ^{235}U nitrate solution in aluminum sphere
7	Water reflected mixed oxide sphere ($^{239}PuO_2$ in U(nat)O_2)
8	Water reflected Pu-U nitrate solution in a cylindrical tank
9	Water reflected Pu-U-Gd nitrate solution in a cylindrical tank
11	Bare 4x4x4 array of 3 kg Pu metal cylinders

Table 4.8: Calculated k_{eff}'s for Additional Computational Benchmark Problems.

Problem id.	PKENO V.a	KENO V.a	MCNP
1	1.0102 ± 0.0045	1.0091 ± 0.0051	1.0138 ± 0.0045
2	1.0131 ± 0.0053	1.0145 ± 0.0049	1.0184 ± 0.0052
3	1.0282 ± 0.0036	1.0219 ± 0.0026	1.0262 ± 0.0023
4	1.0043 ± 0.0046	1.0000 ± 0.0034	1.0001 ± 0.0033
5	0.9998 ± 0.0044	0.9969 ± 0.0044	0.9915 ± 0.0032
6	0.9956 ± 0.0033	0.9969 ± 0.0040	1.0037 ± 0.0032
7	0.7609 ± 0.0041	0.7606 ± 0.0036	0.7520 ± 0.0039
8	1.0203 ± 0.0044	1.0221 ± 0.0031	1.0106 ± 0.0032
9	1.0047 ± 0.0046	0.9982 ± 0.0030	0.9996 ± 0.0045
11†	0.9654 ± 0.0024	0.9586 ± 0.0024	0.9686 ± 0.0014

In addition to the ten benchmark problems, the SHEBA II critical assembly[36] was modeled. SHEBA II is a 304 stainless steel cylindrical vessel with an inside diameter of 47.8 cm. and wall thickness of 1.5 cm. Centered inside the steel vessel is a 304 stainless steel thimble which has an inside diameter of 6.0 cm. and wall thickness of 0.64 cm. The SHEBA II assembly contains low enriched uranyl fluoride solution with a critical height of 42.85 cm. A schematic diagram of the vessel is presented in Figure 4.1.

PKENO V.a was used to model SHEBA II. For comparison, KENO and MCNP were also used to calculate the critical assembly. Each code used 30,000 histories to

†150,000 histories

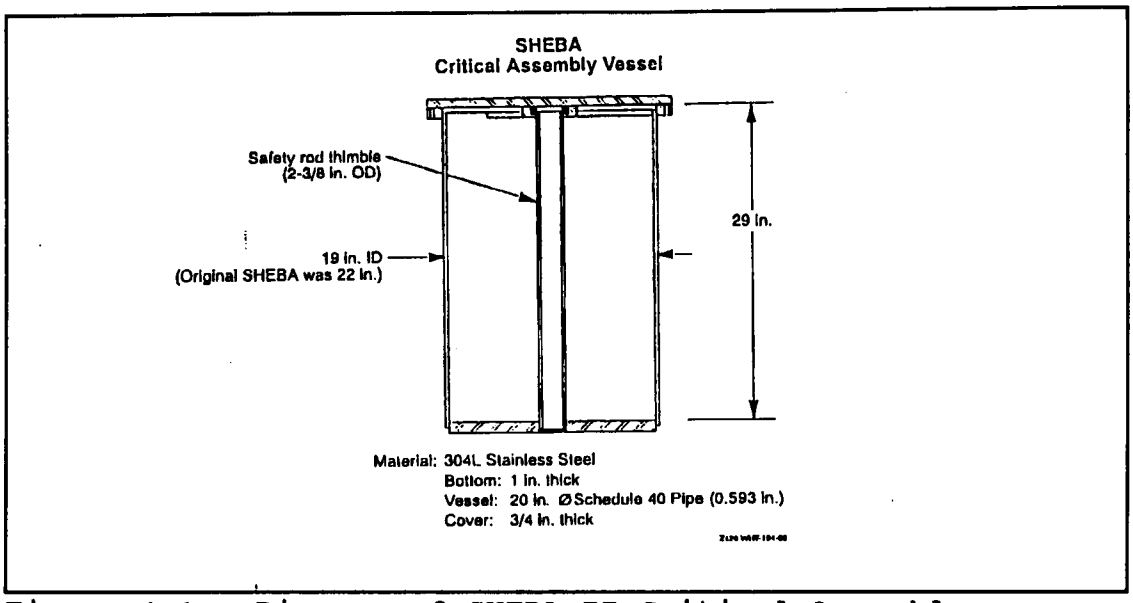


Figure 4.1: Diagram of SHEBA II Critical Assembly.

model the problem. The calculated results obtained with PKENO, MCNP and KENO are 0.9997 $\pm$ 0.0045, 0.9934 $\pm$ 0.0040 and 0.9910 $\pm$ 0.0049, respectively. The multiplication factor obtained with PKENO agrees with MCNP (i.e., within 2 standard deviations). Moreover, the PKENO and MCNP results are within 1.0% of the KENO result.

4.4 Calculational Efficiency

Whenever a new transport code is developed, the code is always benchmarked against experimental data and other transport codes to evaluate the accuracy of the new code. In addition, the speed (i.e., CPU time) in which the code performs the calculation is typically evaluated and ultimately determines the usefulness of the new code. In an attempt to quantify the calculation efficiency of PKENO V.a, the CPU times for the 50 problems presented in Sections 4.1 - 4.3 are compared with KENO V.a and MCNP.

A comparison of the CPU time required for each of the 25 KENO Benchmark problems is presented in Table 4.9. Likewise, the CPU times required for the 14 MCNP benchmark problems and 11 additional benchmark problems are presented in Tables 4.10 and 4.11, respectively. With regard to the times in Table 4.9, the PKENO CPU time is on average 1.8 and 5.3 times slower than MCNP and KENO, respectively. It should be noted that PKENO is faster than MCNP for problem three. PKENO is faster because region and energy dependent biasing of the paraffin reflector was used in the PKENO model. With regard to the cases presented in Table 4.10, PKENO is on average 2.7 and 7.2 times slower than MCNP and KENO, respectively. Examination of the series of problems in Table 4.11 reveals PKENO is an average of 2.1 times slower than MCNP and 5.6 times slower than KENO.

The CPU times for the fifty test problems indicated KENO V.a is faster than MCNP and PKENO. The speed of KENO relative to MCNP is attributed to the KENO geometry package which is computationally more efficient for particle tracking than MCNP's generalized geometry package. In addition, a multigroup collision treatment does not require the selection of the interacting nuclide or type of reaction (i.e., elastic or nonelastic) as a continuous energy collision. Consequently, particle collisions in multigroup KENO are treated much faster than a continuous energy collision treatment in PKENO. Although PKENO is much slower than KENO V.a, PKENO is only a factor of two slower than MCNP for most problems. The longer CPU times required by PKENO are attributed to inefficient handling of the cross section arrays during tracking. For instance, PKENO does not store the pointer location in the energy array for a

particle collision. Consequently, if an energy is encountered repeatedly for a particular nuclide, PKENO must search the energy array for the correct location of the incident energy. In addition, PKENO does not store the mixed total cross section which is used to determine the next collision site. Consequently, if an energy is encountered repeatedly, PKENO must recalculate the mixed total cross section for a mixture at each collision site. These inefficiencies may slow down the tracking procedure during the random walk. Since PKENO utilizes the KENO geometry package, improvements in the calculation efficiency should be possible. Moreover, PKENO should be faster than MCNP for some problems. Despite the inefficiencies, PKENO V.a can be used to accurately perform nuclear criticality evaluations.

Table 4.9: Comparison of CPU Times for the 25 KENO V.a Benchmark Problems.

Problem	PKENO V.a (min.)	KENO V.a (min.)	MCNP (min.)
1	1.56	0.34	0.90
2	1.60	0.30	0.90
3	7.85	2.07	22.02
4	7.63	1.45	2.99
5	7.81	1.51	2.88
6	1.11	0.19	0.41
7	1.54	0.23	0.79
8	1.41	0.19	0.70
9	4.10	0.49	2.45
10	1.60	0.38	0.98
11	0.92	0.19	0.50
12	3.05	0.47	1.57
13	1.60	0.26	0.69
14	1.51	0.30	0.65
15	16.51	3.41	13.37
16	4.54	0.66	2.68
17	3.35	0.81	2.69
18	10.35	2.41	12.76
19	3.09	0.51	1.55
20	2.62	0.51	2.31
21	11.03	4.01	9.47
22	1.88	0.47	1.54
23	1.60	0.34	0.99
24	1.60	0.34	0.94
25	1.59	0.34	0.91

Table 4.10: Comparison of CPU Times for MCNP Benchmark Problems.

Problem	PKENO V.a (min.)	KENO V.a (min.)	MCNP (min.)
ex1	4.10	0.53	1.53
ex2.1	3.63	0.49	1.20
ex2.2	5.65	0.66	1.83
ex2.3	11.47	1.19	4.07
ex3	6.44	1.64	5.40
ex4	19.16	3.84	10.71
ex5	16.43	2.86	12.87
p12	3.84	0.49	1.25
p13	0.90	0.17	0.28
p21	9.37	0.87	2.38
p22	7.30	0.66	1.88
p31	8.51	1.02	2.89
p32	26.73	5.35	9.36
p41	19.63	4.84	13.83

Table 4.11: Comparison of CPU Times for Additional Benchmark Problems.

Problem	PKENO V.a	KENO V.a	MCNP
1	26.33	5.40	8.96
2	18.71	3.67	6.74
3	17.25	3.65	10.78
4	17.17	3.50	11.10
5	1.47	0.23	0.48
6	15.94	3.16	9.82
7	26.65	5.01	9.21
8	22.72	3.82	14.29
9	12.22	1.56	6.48
11	4.33	0.70	3.06
SHEBA	7.68	1.45	5.68

Chapter Five

Conclusions and Recommendations

5.1 Conclusions

KENO V.a is a multigroup Monte Carlo code which is the most widely used production level criticality safety code in the nuclear industry.[26,27] KENO is used extensively in the nuclear community to perform standardized computer analyses of fissile systems. With regard to multigroup cross sections, these data sets are generated from point cross section data by averaging over a specified number of energy groups using a flux weighting spectrum. Consequently, the multigroup cross section set is only valid for a system having a flux spectrum similar to the weighting spectrum. In contrast, a continuous or point energy cross section library can be used for a wide variety of problems. Moreover, continuous energy methods avoid many of the assumptions and approximations inherent in multigroup methods. Since angle and energy are treated as continuous variables, exact energy angle formulas can be used in point Monte Carlo to describe the kinematics of particle collisions. As a result, the continuous energy random walk is more analogous to the actual particle transport. Due to the smaller amount of disk storage and CPU time required for multigroup calculations, multigroup approaches have been preferred over continuous energy approaches in the past to solve the transport equation. With the advent of high performance computers (i.e., dedicated workstations), storage and CPU limitations are less restrictive, thereby making continuous energy

methods viable for transport calculations.

A continuous energy version of the multigroup Monte Carlo code KENO V.a has been developed named PKENO V.a. Since a point cross section library which is based on a continuous energy library such as ENDF/B-V is not available for direct implementation into KENO, PKENO has been modified to utilize an existing point cross section library. Moreover, continuous energy cross sections which are based on ENDF/B-V data are readily available for MCNP. Consequently, the new version of KENO V.a has been developed to perform the random walk using MCNP cross sections.

In order to utilize continuous energy cross sections, extensive modifications have been made to KENO V.a. Several subroutines related to input processing and particle tracking were modified significantly. In addition, twenty-five additional subroutines have been added to the original code package. Although point cross sections are used in the calculation, the user must specify a group structure in order to utilize the current biasing schemes in KENO which are group and region dependent. Despite the extensive modifications, the changes are relatively transparent to the user. As a result, an experienced KENO V.a user can perform a PKENO V.a calculation with only minor modifications to the input.

Regarding the random walk, the primary modifications are performed in subroutine GUIDE which randomly starts the neutrons and controls the tracking for each generation. The spatial starting options in PKENO V.a are the same as in KENO V.a; however, the initial energy selection is made from the continuous energy fission spectrum of the fissionable isotope present in the mixture. If more than one fissionable isotope is

present, PKENO V.a randomly selects the appropriate spectrum to sample. Once the initial source distribution is established, PKENO performs the loop over generations and calls subroutine TRACK which performs the tracking of the individual histories within each generation.

Extensive modifications have been made in TRACK to facilitate the use of point cross sections. In particular, the macroscopic total cross section, Σ_t^{mix}, for a mixture at a particular energy is calculated and used to select the next collision site. Since the collision treatment required significant modifications, a new subroutine named COLLISION has been developed to process the neutron interactions. Once the collision site is determined, a random selection process is used to determine which nuclide interacts with the neutron, and the subsequent collision is modeled. After selecting the interacting nuclide, the particle's weight is reduced by the nonabsorption probability, and the absorption and fission probabilities are calculated. Depending on the particle's weight, splitting and/or Roulette is performed as necessary. It should be noted that a distinction between a nonelastic and elastic reaction is not made in the multigroup collision treatment; however, an explicit identification of the collision type is dictated by the point cross sections. In order to process the collision, a random selection process is used to determine if the collision is elastic or nonelastic. If the collision is nonelastic, the appropriate nonelastic collision is selected and modeled. Based on the collision type, the exiting angle and energy of the neutron is determined. Following the collision treatment, the *FISSION* portion of TRACK is used to process a possible fission reaction. The fission treatment in PKENO is similar to KENO except the energy of the emitted

neutron is sampled from the appropriate continuous energy fission spectrum.

In order to test the new code, PKENO V.a was benchmarked against the multigroup version of KENO V.a and MCNP using fifty test problems. Initially, PKENO was tested using the twenty-five benchmark problem set for KENO V.a. The calculated k_{eff}'s obtained with PKENO agree (i.e., within $\pm 2\sigma$) with the results obtained with KENO and MCNP. Furthermore, twenty of the twenty-five cases are critical problems, and PKENO V.a has a 0.24% average deviation from $k_{eff} = 1.0$ for the twenty problems. In comparison, MCNP and KENO V.a have a 0.10 and 0.36% average deviation from critical, respectively.

After testing the new code against the KENO benchmark problems, PKENO was used to calculate fourteen additional MCNP test problems. As in the previous cases, the results obtained with PKENO are within two standard deviations of the results obtained with MCNP and KENO V.a. Of the fourteen problems, thirteen cases are critical problems, and the PKENO results have an 0.75% average deviation from critical. In comparison, the MCNP and KENO results have an 0.94 and 0.68% average deviation from $k_{eff} = 1.0$. The remaining set of test cases is a series of eleven additional benchmark problems. The calculated results for these problems are also within two standard deviations of the results obtained with MCNP and KENO V.a. The results obtained for the fifty problems demonstrate the ability of PKENO to accurately model a wide variety of fissile systems.

5.2 Recommendations for Future Work

Although a new continuous energy criticality code named PKENO V.a has been developed, additional work needs to be performed to make PKENO a production level criticality safety code. In particular, the MCNP cross sections provide nuclides at various temperatures. If the system temperature is different than the cross section library temperature, PKENO does not adjust the cross sections to account for the temperature differences between the system and library. Consequently, a subroutine should be created which adjusts the cross sections according to temperature differences between the actual system and cross sections. With regard to computing time, the CPU times required for a calculation should be on the order of MCNP. For problems in which the PKENO geometry package has a distinct advantage over generalized geometry, PKENO should actually be faster than MCNP.

The longer CPU times required by PKENO are attributed primarily to inefficient handling of the cross section arrays during tracking. For instance, PKENO does not store the pointer location in the energy array for a particle collision. Also, PKENO does not store the mixed total cross section which is used to determine the next collision site. Consequently, if an energy is encountered repeatedly, PKENO must search the energy array for the correct location of the incident energy and recalculate the mixed total cross section for a mixture at each collision site. Therefore, additional modifications should be made to remove these inefficiencies. Additional coding inefficiencies may be identified by profiling the code and targeting the areas which are CPU intensive. Such improvements should decrease the amount of CPU time required for a calculation.

The problems presented in this work require minimal CPU time (e.g., on the order of minutes). Consequently, the differences in CPU time between PKENO and MCNP and KENO are tolerable for short problems. However, the differences in CPU time are intolerable for more CPU intensive problems (e.g., on the order of hours). Once the coding inefficiencies are removed from PKENO, additional problems which require more CPU time should be calculated with PKENO and compared with MCNP and KENO V.a. Improvements in the coding efficiency should make PKENO a production level criticality safety code.

References

References

1. P. N. Stevens, "NE-581 Shielding Class Notes," Nuclear Engineering Department, University of Tennessee, (1994).

2. M. W. Westfall, "NE-543 Criticality Safety Class Notes," Nuclear Engineering Department, University of Tennessee, (1993).

3. Computational Science Education Project-electronic book, http://csep1.phy.ornl.gov/mc/node1.html, (1995).

4. "MCNP 4A: Monte Carlo N-Particle Transport Code System," Los Alamos National Laboratory LA-12625-M, (November 1993).

5. Ely M. Gelbard, "Overview of the History of Neutronics Computations," *Trans. Am. Nucl. Soc.*, **66**, 240 (1992).

6. Jerome Spanier, "Reflection on Early Monte Carlo Calculations," *Trans. Am. Nucl. Soc.*, **66**, 242 (1992).

7. James J. Duderstadt, Louis J. Hamilton, Nuclear Reactor Analysis, John Wiley & Sons, New York, (1976).

8. "ENDF-102 Data Formats and Procedures for the Evaluated Nuclear Data File ENDF-6," BNL-NCS 44945, Brookhaven National Laboratory, (October 1991).

9. "ENDF-201 ENDF/B-VI Summary Documentation," BNL-NCS-17541, Brookhaven National Laboratory, (October 1991).

10. Charles D. Harmon II, Robert D. Busch, Thomas Hilton Jones, "Comparison of Results from KENO, MCNP, MONK and TWODANT on Twelve Criticality Benchmark Problems," Department of Chemical & Nuclear Engineering, University of New Mexico.

11. H. L. Dodds, "NE-571 Reactor Physics Class Notes," Nuclear Engineering Department, University of Tennessee, (1992).

12. W. Martin and F. Brown, "Monte Carlo Methods for Particle Transport," *Trans. Am. Nucl. Soc.*, **60**, 336 (1989).

13. L. M. Petrie, N. F. Landers, "KENO V.a: An Improved Monte Carlo Criticality Program with Supergrouping," NUREG/CR-0200, Vol. 2, Section F11, ORNL/NUREG/CSD-2/V1/R1, December 1984.

14. "SCALE: A Modular Code System for Performing Standardized Computer Analysis for Licensing Evaluation," NUREG/CR-0200, Vols. 1-5, U. S. Nuclear Regulatory Commission (revisited and reissued January 1990).

15. Todd Postma, "VIM - A Continuous Energy Neutronics and Photon Transport Code,"http://neutrino.nuc.berkley.edu/neutronics/todd/ANECL.codes/vim.html, University of California, Berkley.

16. F. Brown and F. Bischoff, "Computational Geometry for Reactor Applications," *Trans. Am. Nucl. Soc.*, **57**, 112 (1988).

17. B. Basoglu, "Parallel Methods Development for a New Version of the KENO V.a Criticality Safety Code," Ph. D. Dissertation, Nuclear Engineering Department, University of Tennessee, (1995).

18. R. E. MacFarlane, D. W. Muir and R. M. Boicourt, "The NJOY Nuclear Data Processing System Volumes I & II: User's Manual, "LA-9303-M Vol I & II (ENDF-324), Los Alamos National Laboratory, (May 1982).

19. T. M. Sutton and F. Brown, "Parallel Monte Carlo for Reactor Calculations," Proceedings of the Topical Meeting on Advances in Reactor Physics, Vol. 1, 406 (April 1994).

20. F. Brown, "Vectorization of Three-Dimensional General-Geometry Monte Carlo," *Trans. Am. Nucl. Soc.*, **53**, 283 (1986).

21. F. Brown, "Vectorized Monte Carlo Applications in Reactor Physics Analysis," *Trans. Am. Nucl. Soc.*, **46**, 727 (1984).

22. D.C. Wade, M.J. Lineberry, R.E. Prael, "Monte Carlo Analysis of ZPR Heterogeneity," *Trans. Am. Nucl. Soc.*, **21**, 446 (1975).

23. Adelmo S. Pallotta and Roger N. Blomquist, "VIM Monte Carlo Versus CASMO Comparisons for BWR Advanced Fuel Designs," Proceedings of the Topical Meeting on Advances in Reactor Physics, Vol. 1, 107 (April 1994).

24. M. E. Dunn, "Nuclear Criticality Safety Evaluation of the ^{233}U Inventory at the Oak Ridge National Laboratory Using ENDF/B-V Cross Sections," M. S. Thesis, Nuclear Engineering Department, University of Tennessee, (1994).

25. P. N. Stevens, "NE-582 Monte Carlo Methods Class Notes," Nuclear Engineering Department, University of Tennessee, (1995).

26. D. F. Hollenbach, "Development and Evaluation of a Vectorized Version of the Monte Carlo Code KENO V.a," Ph. D. Dissertation Proposal, Nuclear Engineering Department, University of Tennessee, (1991).

27. Gregg W. McKinney, John C. Wagner, James E. Sisolak, "MCNP/KENO Criticality Comparison," Los Alamos National Laboratory.

28. L. M. Petrie, D. F. Hollenbach, LMES ORNL, Personal Communication August 1995.

29. H. L. Dodds, "NE-421 Nuclear Criticality Safety Class Notes," Nuclear Engineering Department, University of Tennessee, (1992).

30. Arthur B. Chilton, J. Kenneth Shultis and Richard E. Faw, Principles of Radiation Shielding, Prentice-Hall Inc., Englewood Cliffs, NJ, (1984)

31. "RSIC Software Directory, A Summary of the Computer Code Collection (CCC), Peripheral Shielding Routines (PSR), and Data Library Collection (DLC)," Oak Ridge National Laboratory, (June 1994).

32. S. Goluoglu, "Three Engineering Practice Project Reports," M. S. Practice School Report, Nuclear Engineering Department, University of Tennessee, (August 1994).

33. D. F. Hollenbach, "Vectorization Methods Development for a New Version of the KENO V.a Criticality Safety Code," Ph. D. Dissertation, Nuclear Engineering Department, University of Tennessee, (August 1992).

34. Charles D. Harmon, Robert D. Busch, Judith F. Briesmeister and R. A. Forster, "Criticality Calculations with MCNP: A Primer," LA-12827-M, Los Alamos National Laboratory, (August 1994).

35. Daniel J. Whalen, David A. Cardon, Jennifer L. Uhle and John S. Hendricks, "MCNP: Neutron Benchmark Problems," LA-12212, Los Alamos National Laboratory, (November 1991).

36. "MCNPDAT MCNP, Version 4, Standard Neutron Cross Section Data LIbrary Based in Part on ENDF/B-V," DLC-105 RSIC Data Library Collection, Oak Ridge National Laboratory (January 1994).

Appendix A

Listing of PKENO V.a Subroutine TRACK

```
      subroutine track(neut,ninf,mixc,impc,mat,imp,igeom,xx, kbnds1,
     * kbnds2,lba,deltx,delty,deltz,fisden,nubank,fsbank,fwrr,lim,
     * tp,tu,th,ta,
     * isggc,lsg,neucnt,vinv,chi,
     * mal,a,alb,wtavg,flux,fleak,fmabs,fmfis,limdim,kboxc,plim,
     * cpol,spol,mwxsc,mwalb,probx,proba,iabsg,iregc,ndxlba,irega,
     * indxx,indxy,indxz,nbxmx,nbymx,nbzmx,stacka,stackh,
     * holx,holy,holz,lholu,khole,ifh,ilh,xtra,
     * nsct1,lensg,nextsg,lreg,nscnt,
     * lenchi,nm,lnchi,npmix,numix,mixchi,egp,nxss,jxss,xss,nsc,
     * nlg,sigt,dens,xsd,sig,inmt,nsmt,nsab,temp,spl,vnuc,vnr,irt,
     * sin,sel)
c
c.....
      common /albnam/rnames(6)
      character*8 rnames
      common /albdat/ intr(6),idalb(6),nbxl(6),iface,nsig,
     * nalb,nang,ng,awt,lprbx,lprba,nagsg,nag1
c.....
c.....
      common /dimen/  tmax,  tbtch,  dwtav, wthigh,  wtlow,    big,
     1         nba,    npb,  nskip,   nbas, nrstrt, numx1d,
     2       nbank,  nxnbk,  nfbnk,  nxfbk,
     3       lbank,  lfbnk, lnubnk, lnextr, lnfbnk,
     4         ngp,   ngp1, ntypst,   nmat,   matt,   nmix,
     5        kmax,  krefm,   nbox,  nboxt,  ngblu,  nucom,
     6      nbxmax, nbymax, nbzmax, nglobl, matdim,  nacom,
     7      maxmix, maximp,  mang,   nimp, numids,    ncs, ntsets,
     8         nsg, nchigp,  nsct,    mix,   mixt,    npl,   pbxs,
     9      maxara, lbalng,  ndelx,  ndely,  ndelz,
     a      numara, nalvls, lnstka,  lstka, lfstka,    iaa,
     b      numhol, nhlvls, lnstkh,  lstkh, lfstkh,    ihh,
     c      mult
c
c       tmax   is time allowed for the problem
c       tbtch  is time allowed for a generation
c       dwtavg is the default value of avg. wt.
c       wthigh is the wt. at which splitting occurs (wthigh*wtavg)
c       wtlow  is the russian roulette threshold (wtlow*wtavg)
c       big    is the maximum path in a void
c
c       nba    is number of generations
c       npb    is number per generation
c       nskip  is number of generations skipped
c       nbas   is the starting generation
c       nrstrt is generations between restarts
c       numx1d is no. of extra 1-d xsecs
c       nbank  is no. of neutrons that can be stored in neutron bank
c       nxnbk  is extra positions per neutron in neutron bank
c       nfbnk  is no. of neutrons that can be stored in fission bank
c       nxfbk  is extra positions per neutron in fission bank
```

```
c
c       lbank  is total positions per neutron in the neutron bank
c       lfbank is total positions per neutron in the fission bank
c       lnubnk is the length of common/nutron/ in the neutron bank
c       lnextr is length of extra data from subroutine extra
c       lnfbnk is the length of common/nutron/ in the fission bank
c       ngp    is number of energy groups
c       ngp1   is no. of energy groups + 1
c       ntypst is start type
c       nmat   is number of mixtures on ice tape
c       matt   is no. of mixtures in the problem
c       nmix   is no. of mixing table entries
c       kmax   is number of geometry regions used
c       krefm  is no. of geometry cards read
c       nbox   is the largest unit number specified in the input data
c       nboxt  is the largest unit number in the problem
c       ngblu  is the global unit number
c       nucom  is the number of units having comments in the geometry
c       nbxmax is number of units in the x dir.
c       nbymax is number of units in the y dir
c       nbzmax is number of units in the z direction
c       nglobl is the global array number
c       maxmix is largest mixture no. in geometry
c       maximp is largest biasing region no. in geometry
c       mang   is no. of angles on xsec tape
c       nimp   is no. of biasing regions
c       numids is the number of bias id's requested
c       ncs    is the number of sets of weights read from cards
c       ntsets is the total number of weight group structures read
c                from cards
c       matdim is nbxmax*nbymax*nbzmax
c       nacom  is the number of arrays with comments in the array data
c       nsg    is the number of supergroups
c       nchigp is the no. of groups the fission spectrum varies with
c       nsct   is the number of scattering angles
c       mix    is the no. of different mixtures to be mixed
c       mixt   is the maximum mixture no. to be mixed
c       npl    is the order of legendre coefficients + 1
c       pbxs   is the threshold for printing messages for errors found
c                in processing the angles and probabilities
c       maxara is the maximum array number
c       lbalng is the length of the composite lba array
c       ndelx  is the length of the composite delx array
c       ndely  is the length of the composite dely array
c       ndelz  is the length of the composite delz array
c       numara is the number of arrays
c       nalvls is the max. depth of nesting for arrays
c       lnstka is the length of stacka (array stack)
c       lstka  is the pointer to stacka within the neutron bank
c       lfstka is the pointer to stacka within the fission bank
c       iaa    is the location in the fission bank of the pointer to
```

```
c                last level in the stacka array iextra(ias)
c        numhol is the number of holes
c        nhlvls is the max. depth of nesting for holes
c        lnstkh is the length of stackh (hole stack)
c        lstkh  is the pointer to stackh within the neutron bank
c        lfstkh is the pointer to stackh within the fission bank
c        ihh    is the location in the fission bank of the pointer to
c                 last level in the stackh array iextra(ihs)
c        mult   is the variable setting the number of storage locations
c                 used by a variable typed double precision
c.....
c.....
      common/final/dtime,diin,diii,igen,iostrt
c.....
c.....
      common /lifetm/ sorwt,tofis,fleakt,sleak,sqleak,fmabst,smabs,
     1 sqabs,fmfist,sfmfis,sqfis,tfisn,gfis,nubar,nusqr,
     2 timg,tmgs,timl,tmls,sqkef,skbar,akbar,rakbar,age,tlf,gbar,gsqr,
     3 self,slfsq,tself,awtsum,awtsq,lvel,lprnt,indxfs
      double precision sorwt,tofis,fleakt,sleak,sqleak,fmabst,smabs,
     1 sqabs,fmfist,sfmfis,sqfis,tfisn,gfis,nubar,nusqr,
     2 timg,tmgs,timl,tmls,sqkef,skbar,akbar,rakbar,age,tlf,gbar,gsqr,
     3 self,slfsq,tself,awtsum,awtsq
c.....gbar   = sum of the average energy group where fission occurs
c.....gsqr   = sum of the squares of the group where fission occurs
c.....nubar  = sum of the average value of nu
c.....nusqr  = sum of the squares of nu
c.....self   = avg. self multiplication of the units
c.....slfsq  = sum of the squares of the self multiplication
c.....indxfs = position of the fission xsec in the extra 1-d xsec array
c.....tfisn  = avg. fission cross section
c.....gfis   = fission weighted energy group
c.....
c.....
      common /logic/  lrun, lplot,  nflx,  nfden,  lmult, newbar,
     1       lunit,  lcku, pmunit, larpos,  lckp, pmapos,
     2       lmhole,  lckh, pmhole,  lhhgh,
     3       lmarry,  lcka, pmarry,  lahgh,
     4       lpaxs,  prt1,  prtp0,  prtap, prtchi,  prtex,
     5        lfa, lcorsp,  lpwt,  lgeom, ldbug,  ltrk,  nadj,
     6       lstgen,  lxtra,  lsgun,  lsav, lstop,  mflag,  lreed,
     7        lnxx,  lsavf,  lrnd,  lsavu, lsavp,  lsavh,  lsava,
     8        mbox,  exrfl,  nesta,  nesth, ppmsg
      logical nflx,nfden,nadj,lpaxs,prtap,prt1,prtp0
      logical lfa,lunit,larpos,lmhole,lmarry,lsgun,lsav
      logical pmapos,pmunit,pmhole,pmarry
      logical lstop,mflag,lreed
      logical prtchi, prtex
      logical ldbug,ltrk
      logical lnxx,lahgh,lhhgh
      logical lsavf,lrnd,lsavu,lsavp,lsavh,lsava,mbox,exrfl
```

```
      logical lpwt,lgeom
      logical lmult,newbar,nesta,nesth,lrun,lplot,ppmsg
      logical lcorsp,lcku,lckp,lckh,lcka,lxtra(2),lstgen
c.....lrun is the execution flag.  a value of false suppresses
c     execution.  default is true.
c.....nadj is the adjoint flag.  default is false.
c.....nflx is the flag for collecting and printing fluxes.
c     default is false.
c.....nfden is the flag for collecting and printing fission densities.
c     default is false.
c.....lfa is the flag for printing fissions and absorptions by region.
c     default is false.
c.....lmult is true if the average self multiplication of a unit is to
c     be calculated.
c.....newbar is true if nu-bar and the average fission group are to be
c     calculated
c.....lunit is the logical key for collecting matrix information by
c     unit type.
c.....lcku is the logical flag for calculating cofactor k-effective
c     by unit type
c.....pmunit is the flag for printing the fission probability matrix by
c     unit type.
c.....larpos is the logical key for collecting matrix information by
c     array position  (location within the array).
c.....lckp is the logical flag for calculating cofactor k-effective by
c     unit location or position in the array.
c.....pmapos is the flag for printing the fission probability matrix by
c     unit location or position in the array.
c.....lmhole is the logical key for collecting matrix information by
c     hole.
c.....lckh is the logical flag for calculating cofactor k-effective by
c     hole.
c.....pmhole is the flag for printing the fission probability matrix by
c     hole.
c.....lmarry is the logical key for collecting matrix information by
c     array.
c.....lcka is the logical flag for calculating cofactor k-effective by
c     array.
c.....pmarry is the flag for printing the fission probability matrix by
c     array.
c.....lpaxs is the print flag to turn on all mixture cross section
c     print options.  if any options are entered in addition to lpaxs,
c     they override lpaxs
c.....prt1 is the print flag for the 1-d cross sections
c.....prtp0 is the print flag for the 2-d cross sections (p0)
c.....prtap is the print flag for the angles and probabilities
c.....prtchi is the print flag for chi, the fission spectrum
c.....prtex is the print flag for the extra 1-d cross sections
c.....lcorsp is true for printing xsec-albedo correspondence arrays
c.....lpwt is logical key for printing weight average
c.....ldbug keys input debug print
```

```
c.....ltrk keys debug tracking print
c.....lgeom is print flag for input geometry data in kenog & rdbox
c.....lplot is the logical flag indicating whether a picture plot
c     is to be generated.  defalult value is false.
c
c.....lahgh is true if matrix information by array is collected
c     at the highest level and false if at the lowest level.
c.....lhhgh is true if matrix information by hole is collected
c     at the highest level and false if at the lowest level.
c.....lxtra holds space for logical input parameters
c.....lstgen is set true upon completion of the last generation that
c     is run.  it is used for restart in the search routines.
c.....lsgun is set true for a single unit problem.
c.....lsav is false if fissions and absorptions by region were written
c     on restart and rstrt=wstrt and the restarted problem says to
c     suppress fissions and absorptions by region.  a warning message
c     is written, fissions and absorptions by region are calculated
c     but not printed
c.....lstop is used to bypass data reading in datain if an end data was
c     encountered before going to datain.
c.....mflag is set true when an error is found, so execution can be
c     terminated when data processing is completed.
c.....lreed is true if a read flag was read when the end param flag
c     was expected.
c.....lnxx is true if an albedo option is to be exercised
c.....lsavf is false if fluxes were written on restart and rstrt=wstrt
c     and the restarted problem says to suppress the fluxes.  a warning
c     message is written and the fluxes are calculated but not printed.
c.....lrnd is true if the restarted problem is to continue with the
c     original random sequence.  to change random sequence, enter a
c     random number as a parameter
c.....lsavu is false if matrix information by unit type was written
c     on the restart unit and rstrt=wstrt and the restarted problem
c     specifies suppressing it.  a warning message is printed and
c     the data is calculated but not printed.
c.....lsavp is false if matrix information by unit position was
c     written on the restart unit and rstrt=wstrt and the restarted
c     problem specifies suppressing it.  a warning message is printed
c     and the data is calculated but not printed.
c.....lsavh is false if matrix information by hole was
c     written on the restart unit and rstrt=wstrt and the restarted
c     problem specifies suppressing it.  a warning message is printed
c     and the data is calculated but not printed.
c.....lsava is false if matrix information by array was
c     written on the restart unit and rstrt=wstrt and the restarted
c     problem specifies suppressing it.  a warning message is printed
c     and the data is calculated but not printed.
c.....mbox is true if multiple units are specified
c.....exrfl is true if a global reflector is present
c.....nesta is logical flag for nested arrays. default=false
c.....nesth is logical flag for nested holes.  default=false
```

```
c.....ppmsg is the flag for printing posit error messages.
c     ppmsg is false while drawing a picture and true while tracking.
c.....
c.....
      common /lowbnd/ l01,l02,l03,l04,l05,l06,l07,l08,l09,l10,
     *               l11,l12,l13,l14,l15,l16,l17,l18,l19,l20
c     l01 is the lower bound for numhol
c     l02 is the lower bound for nalb
c     l03 is the lower bound for nang
c     l04 is the lower bound for lnextr
c     l05 is the lower bound for lbalng
c     l06 is the lower bound for numara and maxara
c     l07 is the lower bound for nalvls
c     l08 is the lower bound for nhlvls
c     l09 is the lower bound for numx1d
c     l10 is the lower bound for nsct1
c     l11 is the lower bound for nsct
c     l12 is the lower bound for ndelx,ndely,ndelz
c     l13 is the lower bound for lprbx
c     l14 is the lower bound for lprba
c.....
c.....
      common /matrx/ nbxgbl,nbygbl,nbzgbl,kglobl,kgore,ngx,ngy,ngz,ng01
c     nbxgbl,nbygbl,nbzgbl are nbxmax,nbymax,nbzmax in the global array
c     kglobl is kcore of the global array
c     ngx,ngy,ngz define the current position in the global array
c     kgore is kcore of the first array level at that position
c     ng01 is the lower index on the global lba
c.....
c.....
      common/nutron/    wt,    tme,     u,     v,     w,
     1       nunum,  kcol,     x,     y,     z,
     2       kcore,   nbx,    nby,   nbz,     k,    ig,    e,nutn,
     3    extra(10),    kr,     ki,    ll,    k1,    k2,  igeo,
     4          x1,    y1,     z1,  xxk1,  xxk2,  xxk3,
     5        xxk4,  xxk5,   xxk6,  xxk7,  xxk8,
     6           m,     n, etausd, nsplit, nfiss,
     7       iself,   ias,    ihs,   now,
     8      imatr1, imatr2, imatr3, imatr4
      dimension iextra(10)
      equivalence (extra(1),iextra(1))
c.....
c.....
      common /titl/ title(20),rdwrd
      character*4 title,rdwrd
c     title is the problem title
c     rdwrd is the block identifier specified on the first data block
c.....
c.....
      common /unit/ inpt, outpt, icexs, albdo, wts, skrt, rstrt, wstrt,
     1 ampxs, direct(5), i0, i1, i2, i3, i4, nspare(4),igs
```

```
      integer outpt,albdo,skrt,rstrt,wstrt,direct,wts,icexs,ampxs
c.....
c.....
c  dunn
c  modification:  added common block for split neutron data
      common /split/ ithn,sigtith,sige,sigins,sigam,siga,signp,sigf,
     1  ltherm,ji,je,ei1,ei2,ee1,ee2,nsi,leel,lein,neel,nein,selith,
     2  sinith,neutn
c.....
c.....
c  dunn
c
c  dunn
c  modification:  removed following dimension statement--not needed
c      dimension fsp(limdim),ang(limdim,l11:nsct),prob(limdim,l10:nsct1)
c  dunn
c
c******fsbank contains x,y,z,nbx,nby,nbz,k,energy gp (nsig),e **********
c  dunn
c  modification: removed mwa dimension statement--not needed
c      dimension mwa(3,lensg,matt)
c  dunn
c
c  dunn
c  modification:  removed fap,fnap,fnfp from following dimension
c                 statement.  also changed chi to 1-d array
c      dimension xx(7,krefm),fap(lensg,matt),mat(krefm),
c     *          fwrr(nfbnk),nubank(lbank,nbank),fsbank(lfbnk,nfbnk),
c     *          neucnt(nsg),lim(nsg,3),lsg(ngp),
c     *          chi(ngp,nchigp,matt),fnap(lensg,matt),fnfp(lensg,matt),
c     *          isggc(ngp),kboxc(krefm)
      dimension xx(7,krefm),mat(krefm),
     *          fwrr(nfbnk),nubank(lbank,nbank),fsbank(lfbnk,nfbnk),
     *          neucnt(nsg),lim(nsg,3),lsg(ngp),
     *          chi(lnchi),
     *          isggc(ngp),kboxc(krefm)
c  modification:  removed sigt from following dimension statement,
c                 dimension sigt elsewhere
c      dimension fmabs(lensg,lreg),fmfis(lensg,lreg),fleak(lensg),
c     *          igeom(krefm),kbnds1(nboxt),kbnds2(nboxt),
c     *          wtavg(lensg,nimp),sigt(lensg,matt),vinv(lensg),
c     *          deltx(l12:ndelx),delty(l12:ndely),deltz(l12:ndelz),
c     *          mixc(maxmix),impc(maximp),imp(krefm),
c     *          flux(lensg,kmax),fisden(kmax,3)
      dimension fmabs(lensg,lreg),fmfis(lensg,lreg),fleak(lensg),
     *          igeom(krefm),kbnds1(nboxt),kbnds2(nboxt),
     *          wtavg(lensg,nimp),vinv(lensg),
     *          deltx(l12:ndelx),delty(l12:ndely),deltz(l12:ndelz),
     *          mixc(maxmix),impc(maximp),imp(krefm),
     *          flux(lensg,kmax),fisden(kmax,3)
c  dunn
```

```
c
      dimension tp(matdim,matdim),tu(nboxt,nboxt)
      dimension th(l01:numhol,l01:numhol),ta(l06:numara,l06:numara)
      dimension mal(3,l03:nang,lensg)
      dimension alb(l02:iabsg,l02:nalb),a(l03:nang,l02:iabsg,l02:nalb)
      dimension plim(l03:nang,l02:nalb),cpol(l03:nang,l02:nalb)
      dimension spol(l03:nang,l02:nalb)
      dimension mwxsc(ngp,3),mwalb(l02:ng,3)
      dimension probx(l13:lprbx),proba(l14:lprba)
      dimension iregc(krefm)
c
c dunn
c modification:  changed following dimension statment no x1d xsec.
c       dimension x1d(lensg,matt,l09:numx1d),xtra(l04:lnextr)
      dimension xtra(l04:lnextr)
c dunn
c
      dimension ndxlba(l06:numara),irega(l06:maxara),
     *          indxx(l06:numara),indxy(l06:numara),indxz(l06:numara),
     *          nbxmx(l06:maxara),nbymx(l06:maxara),nbzmx(l06:maxara)
c     ndxlba holds the starting location of each lba array
c     irega is the array correspondence array
c     indxx,indxy,indxz are offsets into deltx,delty,deltz
c     nbxmx,nbymx,nbzmx contain nbxmax,nbymax,nbzmax for each array
      dimension sold(3),snew(3),s1old(3)
      dimension  holx(l01:numhol), holy(l01:numhol), holz(l01:numhol)
      dimension khole(l01:numhol),lholu(l01:numhol)
      dimension ifh(krefm),ilh(krefm)
c     sold is the beginning point of the track
c     snew is the crossing point
c     s1old is the end point of the track
c     holx, holy, holz are the x,y&z offset of the hole
c     lholu is the unit within the hole
c     khole is the region that contains the hole
c     ifh and ilh are the first and last holes in the region
      dimension stacka(4,l07:nalvls),stackh(l08:nhlvls)
c
c  dunn
c  modification: added dimension statement
c     mixchi stores a 1 if mixture has a chi spectrum or 0 if not.
      dimension lenchi(nm),npmix(2*numix),mixchi(maxmix),egp(ngp1),
     *nxss(16,nm+nsc),jxss(32,nm+nsc),xss(nlg,nm+nsc),
     *dens(nm),sigt(nm),xsd(nm+nsc,9),sig(66),inmt(66),nsmt(nsc),
     *nsab(nm),temp(nm),spl(23,nbank),sin(nm),sel(nm)
c  dunn
c
      integer stacka,stackh
      dimension uvw(3),nxyz(3),ih(6),xxedge(3),irev(6),logflg(6)
      equivalence (u,uvw(1)),(nbx,nxyz(1)),(x,xxedge(1))
      equivalence (logxp,logflg(1)),(logxm,logflg(2)),(logyp,logflg(3)),
     *          (logym,logflg(4)),(logzp,logflg(5)),(logzm,logflg(6))
```

```
      logical logflg,logper,lalb
      logical larray,ic,lab,logxp,logxm,logyp,logym,logzp,logzm,
     *        lsplit,lrghol,lhole,lcros,lnmhol
c     lrghol logical flag is true if the region contains holes
c     lhole  logical flag is true if you are in a hole
c     lnmhol logical flag is true if the problem utilizes holes
      integer lba(105:lbalng)
      equivalence(fself,lself)
      logical lself, lglobl
      logical matrix
      double precision fleak, fmabs, fmfis, psi, toosml
      double precision fltrn, sflra, exprn
      parameter ( toosml = 1.0d-10 )
      data ih/1,1,2,2,3,3/
      data irev/2,1,4,3,6,5/
c
      if (nalb.gt.0) call  albin(mal,alb,a,plim,cpol,spol,mwxsc,mwalb,
     *                           probx,proba,ngp,iabsg)
      if (ldbug) call  ldwrt(kmax,kmax1,limdim,neut,nsct,nsct1,lensg,
     *                       matt,fsp,ang,prob,mwa,fap,fnap,fnfp,sigt)
      isc    = 0
c     set local flag for matrix calculation
      matrix = larpos .or. lunit .or. lmhole .or. lmarry
      lnmhol = numhol.gt.0
      lself  = .false.
  100 if (neut.eq.0) return
c*****************************start fstart ****************************
      call  move(wt,nubank(1,neut),lnubnk)
      call move(ithn,spl(1,neut),23)
      if (nesta) call  move(stacka,nubank(lstka,neut),lnstka)
      if (nesth) call  move(stackh,nubank(lstkh,neut),lnstkh)
      now    = nglobl
      lsthol = 0
      if (kcore.gt.0) now    = mat(kcore)
      if (larpos)     kft    = iextra(imatr1)
      if (lunit )     lls    = iextra(imatr2)
      if (lmhole)     nhl    = iextra(imatr3)
      if (lmarry)     nar    = iextra(imatr4)
c
c     set larray and lhole here
c
      ll     = kboxc(k)
      larray = .not.nbx.eq.0
      nhole  = 0
      if (lnmhol) nhole  = iextra(1)
      lhole  = nhole.gt.0
      lglobl = .false.
      if (larray) then
        nbxmax = nbxmx(now)
        nbymax = nbymx(now)
        nbzmax = nbzmx(now)
```

```
        lglobl = now.eq.nglobl .or. .not.exrfl
        if (nesta.and..not.lglobl) then
          if (iextra(ias).gt.0) then
            kcor   = stacka(1,1)
            if (kcor.gt.0) lglobl = lglobl .or. mat(kcor).eq.nglobl
          end if
        end if
      end if
c     set the position in the global array
      if (larpos .and. larray) then
        if (now.eq.nglobl) then
          call  move(kgore,kcore,4)
        else if (nesta) then
          call  move(kgore,stacka(1,1),4)
        end if
      end if
c     if arrays are placed in holes in the global unit,(there is
c     no core boundary card defining a global reflector), then the
c     global unit must be last (the nboxt unit)
c*******nsig is the position of the group in the supergroup corresponding
c******* to energy group ig
      if (lmult) fself   = extra(iself)
      ic     = .true.
c**********if nunum is negative, set the splitting flag ******************
      lsplit = nunum.lt.0
      nunum  = abs(nunum)
      lalb   = kcol.lt.0
      rpth   = 0.0
      nsig   = isggc(ig)
      if (kcol.eq.0.0) kcol   = impc(imp(k))
      neut   = neut-1
      if (neut.le.144.and.igen.ge.8) then
        neut=neut
      endif
      k1     = kbnds1(ll)
      k2     = kbnds2(ll)
c*******if lsplit is true, the neutron was split and must now be collided
      if (ltrk) call  trkwrt('start  ',nsig,nextsg)
      if (.not.lalb) go to 130
c.....play russian roulette
      kcol   = abs(kcol)
      if (wt.lt.wtlow*wtavg(nsig,kcol)) then
        if (fltrn()*wtavg(nsig,kcol).gt.wt) go to 1220
        wt     = wtavg(nsig,kcol)
      end if
      if ( k.eq.k1 ) go to 270
      go to 140
  130 continue
      if (.not.lsplit) go to 140
      kr     = mixc(mat(k))
      ki     = impc(imp(k))
```

```
      kcol   = ki
      lab    = .false.
      ab     = 0.0
      fisw   = 0.0
      g      = 0.0
      fisn   = 0.0
c
c  dunn
c  modification:  making collision a subroutine invalidates the
c                 following go to statement--i.e. performing splitting
c                 and russian roulette.  therefore, set a flag lfstart=1
c                 and go to 930 which calls collision. upon entering
c                 collision, the flag will indicate go to 960 in
c                 the subroutine and perform the above tasks.
c       go to 960
      lfstart=1
      go to 930
c  dunn
c
c*********************************end fstart **********************************
c
c
c*********************************start path***********************************
  140 if (e.lt.1.31e-7.and.e.gt.1.3e-7.and.neut.eq.229)then
        e=e
      endif
c  140 if (rpth.le.0.0) rpth = exprn(0)
      if (rpth.le.0.0) rpth = exprn(0)
      lrghol = ifh(k).gt.0
      kr     = mat(k)
      if (kr) 160,150,160
  150 pth    = big
      go to 170
  160 kr     = mixc(kr)
c
c  dunn
c  modification:  determine mixed total cross section for mixture kr
      call clear (sigt,nm)
      call clear (sin,nm)
      call clear (sel,nm)
      call clear (ithn,23)
      call mixst (npmix,sigtm,e,nxss,jxss,xss,dens,kr,numix,nm,nsc,nlg,
     *            nsmt,nsab,sigt,sin,sel,ji,je,ei1,ei2,ee1,ee2)
c  dunn
c
c  dunn
c  modification: calculate pth with sigtm at energy e
c       pth    = rpth/sigt(nsig,kr)
      pth    = rpth/sigtm
c  dunn
c
```

```
 170 x1     = x+pth*u
     y1     = y+pth*v
     z1     = z+pth*w
     if (x.eq.x1 .and. y.eq.y1 .and. z.eq.z1) go to 930
     if (ltrk) call  trkwrt('path    ',nsig,nsg)
c*****************************end path*********************************
c
c
c*****************************start inward crossing ******************
     if (k.eq.k1) go to 200
 180 if (ic) go to 190
     ic     = .true.
     if (lrghol) go to 190
     go to 450
 190 km1    = k-1
     call  move(xxk1,xx(1,km1),7)
     igeo   = abs(igeom(k-1))
     n      = -1
     if (lrghol) call  move(sold,x,3)
     etamin = 1.0
     if (ltrk) call  trkwrt('inward  ',nsig,nextsg)
     call  cros
c****** m=0 means a crossing did not occur
     if (m.eq.0) go to 210
     lcros  = .true.
     etamin = etausd
     ki     = km1
     if (.not.lrghol) go to 260
     call  move(snew,x,3)
     go to 230
 200 if (.not.lrghol) go to 450
     call  move(sold,x,3)
     etamin = 1.0
     go to 220
 210 if (.not.lrghol) go to 450
 220 lcros  = .false.
     call  move(snew,sold,3)
 230 is     = ifh(k)
     ie     = ilh(k)
     call  move(s1old,x1,3)
     mhole  = 0
     do 240 ik=is,ie
     if (ik.eq.lsthol) go to 240
     x      = sold(1)-holx(ik)
     y      = sold(2)-holy(ik)
     z      = sold(3)-holz(ik)
     x1     = s1old(1)-holx(ik)
     y1     = s1old(2)-holy(ik)
     z1     = s1old(3)-holz(ik)
     kh1    = kbnds2(lholu(ik))
     call  move(xxk1,xx(1,kh1),7)
```

```
      igeo   = abs(igeom(kh1))
      n      = -1
      if (ltrk) call  trkwrt('hole    ',nsig,nextsg)
      call  cros
      if (m.eq.0) go to 240
      if (etausd.ge.etamin) go to 240
      ki     = kh1
      lcros  = .true.
      etamin = etausd
      mhole  = ik
      call  move(snew,x,3)
  240 continue
      lsthol = 0
      call  move(x,snew,3)
      if (.not.lcros) go to 440
      if (mhole.eq.0) go to 260
      ll     = lholu(mhole)
      k1     = kbnds1(ll)
      k2     = kbnds2(ll)
      if (.not.nesth .or. .not.lhole) go to 250
      iextra(ihs) = iextra(ihs)+1
      stackh(iextra(ihs)) = nhole
  250 nhole  = mhole
      lhole  = .true.
  260 continue
c
      if (nflx) flux(nsig,iregc(k)) = flux(nsig,iregc(k))+
     *  wt*pth*etamin
      k      = ki
      tme    = tme + pth*etamin*vinv(nsig)
      if (kr.gt.0) rpth = rpth*( 1.0 - etamin )
c*****************************end inward crossing **********************
      if (k.gt.k1) go to 140
c*****************************start finbox**************************
  270 continue
      if (igeom(k).ne.-1) go to 140
      if (ltrk) call  trkwrt('finbox  ',nsig,nextsg)
      if (.not.larray) go to 280
      iextra(ias) = iextra(ias) + 1
      call  move(stacka(1,iextra(ias)),kcore,4)
  280 continue
      larray = .true.
      ic     = .true.
      kcore  = k
      now    = mat(k)
      lglobl = lglobl .or. now.eq.nglobl
      nbxmax = nbxmx(now)
      nbymax = nbymx(now)
      nbzmax = nbzmx(now)
      ixof   = indxx(irega(now))
      iyof   = indxy(irega(now))
```

```
      izof   = indxz(irega(now))
      jx     = 2
      jy     = 4
      jz     = 6
      difx   = 0.0
      dify   = 0.0
      difz   = 0.0
      if (x.ne.xx(1,k)) go to 290
      jx     = 1
      nbx    = nbxmax
      go to 330
  290 if (x.ne.xx(2,k)) go to 300
      nbx    = 1
      go to 330
  300 difx   = x - xx(2,k)
      do 310 ii=2,nbxmax
        if (difx.lt.deltx(ii+ixof)) go to 320
  310   continue
      ii     = nbxmax+1
  320 nbx    = ii-1
      difx   = difx-deltx(nbx+ixof)
  330 if (y.ne.xx(3,k)) go to 340
      jy     = 3
      nby    = nbymax
      go to 380
  340 if (y.ne.xx(4,k)) go to 350
      nby    = 1
      go to 380
  350 dify   = y -xx(4,k)
      do 360 ii=2,nbymax
        if (dify.lt.delty(ii+iyof)) go to 370
  360   continue
      ii     = nbymax+1
  370 nby    = ii-1
      dify   = dify-delty(nby+iyof)
  380 if (z.ne.xx(5,k)) go to 390
      jz     = 5
      nbz    = nbzmax
      go to 430
  390 if (z.ne.xx(6,k)) go to 400
      nbz    = 1
      go to 430
  400 difz   = z - xx(6,k)
      do 410 ii=2,nbzmax
        if (difz.lt.deltz(ii+izof)) go to 420
  410   continue
      ii     = nbzmax+1
  420 nbz    = ii-1
      difz   = difz-deltz(nbz+izof)
  430 ll     = 1
      if (mbox) ll = locbox(lba(ndxlba(irega(now))),nbx,nby,nbz)
```

```
      k1     = kbnds1(ll)
      k2     = kbnds2(ll)
      k      = k2
      x      = difx+xx(jx,k2)
      y      = dify+xx(jy,k2)
      z      = difz+xx(jz,k2)
c     set the position in the global array
      if (.not.larpos.or.now.ne.nglobl) go to 270
      call  move(kgore,kcore,4)
      go to 270
c*****************************end finbox*************************
  440 call  move(x1,s1old,3)
c*****************************start posit************************
  450 igeo   = abs(igeom(k))
      go to (470,460,480,470,490,500,510,560,560,560,560,560,560,560,
     *       560,560,560,560,560),igeo
      write(outpt,10100) x,y,z,k1,k2,k,igeo
      stop 134
  460 rsq    = (x1-xx(5,k))**2 + (y1-xx(6,k))**2
      if (z1.le.xx(2,k).and.z1.ge.xx(3,k).and.rsq.le.xx(4,k)) go to 930
      go to 630
  470 if (x1.le.xx(1,k).and.x1.ge.xx(2,k).and.y1.le.xx(3,k).and.y1.ge.xx
     *(4,k).and.z1.le.xx(5,k).and.z1.ge.xx(6,k)) go to 930
      go to 630
  480 rsq    = (x1-xx(4,k))**2 + (y1-xx(5,k))**2 + (z1-xx(6,k))**2
      if (rsq.le.xx(2,k)) go to 930
      go to 630
  490 rsqx   = (y1-xx(5,k))**2 + (z1-xx(6,k))**2
      if (x1.le.xx(2,k).and.x1.ge.xx(3,k).and.rsqx.le.xx(4,k)) go to 930
      go to 630
  500 rsqy   = (x1-xx(5,k))**2 + (z1-xx(6,k))**2
      if (y1.le.xx(2,k).and.y1.ge.xx(3,k).and.rsqy.le.xx(4,k)) go to 930
      go to 630
  510 sgn    = sign(1.0,xx(3,k))
      nhsph  = abs(xx(3,k))
      go to (520,520,530,530,540,540),nhsph
  520 tz1    = sgn*(x1-xx(4,k))
      go to 550
  530 tz1    = sgn*(y1-xx(5,k))
      go to 550
  540 tz1    = sgn*(z1-xx(6,k))
  550 rsq    = (x1-xx(4,k))**2 + (y1-xx(5,k))**2 + (z1-xx(6,k))**2
      if (rsq.le.xx(2,k).and.tz1.ge.-xx(7,k)) go to 930
      go to 630
  560 nhcyl  = (igeo-6)/2
      sgn    = 2*mod(igeo,2)-1
      go to (570,580,590,600,610,620),nhcyl
  570 if (sgn*(x1-xx(5,k)).lt.-xx(7,k)) go to 630
      go to 460
  580 if (sgn*(y1-xx(6,k)).lt.-xx(7,k)) go to 630
      go to 460
```

```
 590 if (sgn*(z1-xx(6,k)).lt.-xx(7,k)) go to 630
     go to 490
 600 if (sgn*(y1-xx(5,k)).lt.-xx(7,k)) go to 630
     go to 490
 610 if (sgn*(z1-xx(6,k)).lt.-xx(7,k)) go to 630
     go to 500
 620 if (sgn*(x1-xx(5,k)).lt.-xx(7,k)) go to 630
     go to 500
c*****************************end posit***************************
c
c
c***************************** start outward crossing ******************
 630 call  move(xxk1,xx(1,k),7)
     n     = 1
     if (ltrk) call  trkwrt('outward ',nsig,nextsg)
     call  cros
     if (nflx) flux(nsig,iregc(k)) = flux(nsig,iregc(k))+
    *  wt*pth*etausd
c
     tme   = tme + pth*etausd*vinv(nsig)
     if (kr.gt.0) rpth = rpth*( 1.0 - etausd )
     k     = k+1
     if (k.le.k2) go to 830
     k     = k-1
     if (ll.eq.ngblu) go to 840
c***************************** end outward crossing ********************
c
c
c*****************************start array*********************************
     if (lhole.and.ll.eq.lholu(nhole)) go to 720
     if (ltrk) call  trkwrt('array   ',nsig,nextsg)
     if (lmult) lself  = .false.
     ic    = .true.
     ifx   = 0
     ify   = 0
     ifz   = 0
     logxp  = .false.
     logxm  = .false.
     if (x.ne.xxk1)    go to 640
     if (u.le.0.0)     go to 640
     ifx   = 2
     nbx   = nbx+1
     logxp  = nbx.gt.nbxmax
     if (logxp) go to 650
     x     = xxk2
     go to 650
 640 if (x.ne.xxk2)    go to 650
     if (u.ge.0.0)     go to 650
     ifx   = 1
     nbx   = nbx-1
     logxm  = nbx.lt.1
```

```
      if (logxm) go to 650
      x      = xxk1
  650 logyp  = .false.
      logym  = .false.
      if (y.ne.xxk3)     go to 660
      if (v.le.0.0)      go to 660
      ify    = 4
      nby    = nby+1
      logyp  = nby.gt.nbymax
      if (logyp) go to 670
      y      = xxk4
      go to 670
  660 if (y.ne.xxk4)     go to 670
      if (v.ge.0.0)      go to 670
      ify    = 3
      nby    = nby-1
      logym  = nby.lt.1
      if (logym) go to 670
      y      = xxk3
  670 logzp  = .false.
      logzm  = .false.
      if (z.ne.xxk5)     go to 680
      if (w.le.0.0)      go to 680
      ifz    = 6
      nbz    = nbz+1
      logzp  = nbz.gt.nbzmax
      if (logzp) go to 690
      z      = xxk6
      go to 690
  680 if (z.ne.xxk6)     go to 690
      if (w.ge.0.0)      go to 690
      ifz    = 5
      nbz    = nbz-1
      logzm  = nbz.lt.1
      if (logzm) go to 690
      z      = xxk5
c     set position in the global array
  690 continue
      if (.not.larpos.or.now.ne.nglobl) go to 700
      call  move(kgore,kcore,4)
  700 continue
      if (logxp .or. logxm .or. logyp .or.
     *    logym .or. logzp .or. logzm) go to 730
c
      lll    = ll
      if (mbox) lll    = locbox(lba(ndxlba(irega(now))),nbx,nby,nbz)
      if (lll.eq.ll)     go to 710
      ll     = lll
      k1     = kbnds1(ll)
      k2     = kbnds2(ll)
      x      = x - (xxk2-xx(2,k2))
```

```
      y      = y - (xxk4-xx(4,k2))
      z      = z - (xxk6-xx(6,k2))
      if (ifx.gt.0) x      = xx(ifx,k2)
      if (ify.gt.0) y      = xx(ify,k2)
      if (ifz.gt.0) z      = xx(ifz,k2)
  710 k      = k2
      if (k.eq.k1) go to 270
      go to 140
c*********************************end array**********************************
  720 k      = khole(nhole)
      x      = x+holx(nhole)
      y      = y+holy(nhole)
      z      = z+holz(nhole)
      ll     = kboxc(k)
      k1     = kbnds1(ll)
      k2     = kbnds2(ll)
      ic     = .true.
      lhole  = .false.
      lsthol = nhole
      nhole  = 0
      if (.not.nesth) go to 140
      lhole  = iextra(ihs).gt.0
      if (.not.lhole) go to 140
      nhole  = stackh(iextra(ihs))
      iextra(ihs) = iextra(ihs)-1
      go to 140
  730 if (now.eq.nglobl) lglobl = .false.
      if (now.eq.nglobl.and..not.exrfl) go to 840
      ll     = kboxc(kcore)
      k1     = kbnds1(ll)
      k2     = kbnds2(ll)
      ira    = irega(now)
      k      = k1
      igeo   = abs(igeom(k))
      kr     = 0
      kcore  = 0
      now    = 0
      larray = .false.
      if (logxm) go to 740
      if (logxp) go to 750
      x      = x - xxk2 + deltx(nbx+indxx(ira)) + xx(2,k1)
      go to 760
  740 x      = xx(2,k1)
      go to 760
  750 x      = xx(1,k1)
  760 if (logym) go to 770
      if (logyp) go to 780
      y      = y - xxk4 + delty(nby+indxy(ira)) + xx(4,k1)
      go to 790
  770 y      = xx(4,k1)
      go to 790
```

```
 780 y      = xx(3,k1)
 790 if (logzm) go to 800
     if (logzp) go to 810
     z      = z - xxk6 + deltz(nbz+indxz(ira)) + xx(6,k1)
     go to 820
 800 z      = xx(6,k1)
     go to 820
 810 z      = xx(5,k1)
     go to 820
 820 ic     = .false.
     x1     = x+big*u
     y1     = y+big*v
     z1     = z+big*w
     nbx    = 0
     if (.not.nesta) go to 630
     if (iextra(ias).eq.0)  go to 630
     call  move(kcore,stacka(1,iextra(ias)),4)
     now    = nglobl
     if (kcore.ne.0) now = mat(kcore)
     larray = .true.
     nbxmax = nbxmx(now)
     nbymax = nbymx(now)
     nbzmax = nbzmx(now)
     iextra(ias) = iextra(ias)-1
     go to 630
 830 ic     = .false.
     go to 140
c*********************albedo treatment************************************
c********if lcor=false neutron is within the core************************
 840 if (.not.lnxx) go to 920
     if (ltrk) call  trkwrt('albedo  ',nsig,nextsg)
     logper = .false.
     do 880 iface=1,6
c.......logflg is true if the neutron is on the face and the direction
c.......cosine allows movement out the face.
        logflg(iface) = xxedge(ih(iface)).eq.xx(iface,k) .and.
    *                   (2*mod(iface,2)-1)*uvw(ih(iface)).gt.0.0
        if (.not.logflg(iface)) go to 880
        ib     = ih(iface)
        ikind  = intr(iface)+1
        ic     = .true.
        go to (920,850,860,870),ikind
c.....mirror or specular reflection
 850    uvw(ib)  = -uvw(ib)
        nxyz(ib) = nbxl(irev(iface))
        go to 880
c.....periodic reflection
 860    nxyz(ib) = nbxl(iface)
        logper = .true.
        go to 880
c.....differential albedo treatment
```

```
 870   call albedo
     * (mal,alb,a,plim,cpol,spol,mwxsc,mwalb,probx,proba,ngp,iabsg)
       nxyz(ib) = nbxl(irev(iface))
       rpth   = 0.0
 880   continue
      if (logper) then
        if (.not.exrfl) then
          ll    = 1
          if (mbox)  ll = locbox(lba(ndxlba(irega(now))),nbx,nby,nbz)
        end if
        k1     = kbnds1(ll)
        k2     = kbnds2(ll)
        k      = k2
        x      = x+xx(2,k2)-xxk2
        y      = y+xx(4,k2)-xxk4
        z      = z+xx(6,k2)-xxk6
        do 900 iface=1,6
          if (logflg(iface)) then
            if (intr(iface).ne.2) then
              xxedge(ih(iface)) = xx(iface,k2)
            else
              xxedge(ih(iface)) = xx(irev(iface),k2)
            end if
          end if
 900    continue
      end if
      if (exrfl) nbx    = 0
      if (now.eq.nglobl.and..not.exrfl) call  move(kgore,kcore,4)
      lglobl = now.eq.nglobl .and. .not.exrfl
      kcol   = -kcol
      newsg  = lsg(ig)
      if (newsg.ne.nextsg) go to 1210
      nsig   = isggc(ig)
c.....play russian roulette
      kcol   = abs(kcol)
      if (wt.lt.wtlow*wtavg(nsig,kcol)) then
        if (fltrn()*wtavg(nsig,kcol).gt.wt) go to 1220
        wt     = wtavg(nsig,kcol)
      end if
      if (k.eq.k1) go to 270
      go to 140
 920  if (igen.gt.nskip) then
        fleak(nsig) = fleak(nsig)+wt
      end if
      go to 1220
c*************************************************************************
c  dunn
c  modification:  changed collision section of track to a subroutine
c*************************************************************************
 930   call collision(lfstart,iregc,flux,pth,igold,rpth,lab,ic,
     * ab,fisw,fap,fnfp,fnap,fisn,g,lself,impc,imp,matrix,
```

```
     * tp,kft,tu,lls,larray,nar,irega,iextra,ta,lhole,
     * nhl,nhole,stackh,th,wtavg,ninf,neut,nscnt,lnmhol,
     * nubank,ignew,nsct1,lreturn,lglobl,
     * mat,vinv,lensg,stacka,
     * npmix,nxss,jxss,xss,dens,numix,nm,nsc,nlg,sigt,xsd,enew,sig,
     * inmt,egp,ith,chi,lenchi,lnchi,lfisflg,mt,nsmt,nsab,temp,spl,
     * vnuc,vnr,irt,sin,sel,ji,je,ei1,ei2,ee1,ee2)
c
c modification: if lreturn=1, then return to guide
      if (lreturn.eq.1) return
c
c*******************************************************************************
c  dunn ---end collision treatment modification
c*******************************************************************************
      igold  = nsig
      if (igen.le.nskip) go to 1120
      kt     = 1
      if (lfa) kt     = iregc(k)
      fmabs(igold,kt) = fmabs(igold,kt)+ab
      fmfis(igold,kt) = fmfis(igold,kt)+fisw
      if (nfden) fisden(iregc(k),1) = fisden(iregc(k),1)+fisw
 1120 if (fisw.le.0) go to 1200
      tofis  = tofis+fisw
      age    = age+fisw*tme
      if (.not.newbar) go to 1130
      tfisn  = tfisn+fisn
      gfis   = gfis+g
c******************************* start fission ******************************
c***** this section loads the fission bank
 1130 rnum   = fltrn()
      fwr    = fisw/rnum
      if (rakbar.gt.fwr) go to 1200
      if (fisw.gt.rakbar) fwr    = rakbar/rnum
      if (nfiss.lt.nfbnk) go to 1150
      ifl    = 1
      rlow   = fwrr(1)
      do 1140 ie=2,nfbnk
        if (fwrr(ie).ge.rlow) go to 1140
        ifl    = ie
        rlow   = fwrr(ifl)
 1140   continue
      if (fwr.le.rlow) go to 1190
      go to 1160
 1150 ifl    = nfiss+1
 1160 continue
      fwrr(ifl) = fwr
      nfiss  = nfiss+1
      if (lnmhol) iextra(1) = nhole
c
c  dunn
c  modification:  changed selection of new energy after fission
```

```
c                to be from specified endf law.
c     igchi  = 1
c***** pick igchi from a vector when available
c     r      = fltrn()
c     do 1170 ig=1,ngp
c       if (r.le.chi(ig,igchi,kr)) go to 1180
c 1170   continue
c*****put in an error message later
c     ig     = ngp
c
c*****select new energy from specified law
      lmt = jxss(3,ith)
c     if (lfisflg.ne.1) then
      do 1165 ii=1,nxss(4,ith)
        mtn =  xss(lmt+ii-1,ith)
        if (mtn.eq.18.or.mtn.eq.19.or.mtn.eq.20.
     *      or.mtn.eq.21.or.mtn.eq.38) go to 1175
 1165   continue
c     else
c     do 1170 ii=1,nxss(4,ith)
c       mtn =  xss(lmt+ii-1,ith)
c       if (mtn.eq.mt) go to 1175
c1170  continue
c     endif
1175  loccmt = xss(jxss(10,ith)+ii-1,ith)
      lengy  = jxss(11,ith)+loccmt-1
      le     = jxss(1,ith)
      ldis   = jxss(11,ith)
      if (nxss(5,ith).ne.0) then
         call sellaw (xss(le,ith),lengy,loccmt,ldis,law,llaw,e)
      else
         law=0
      endif
      call endflaw (xss(le,ith),law,llaw,e,eout,ldis,chi,lenchi,
     *              lnchi,egp,ith)
      e      = eout
c*****determine the group for fission energy
      if(e.ge.egp(1))ig=1
      if(e.le.egp(ngp1))ig=ngp
      do 1178 ii=1,ngp
        if (e.lt.egp(ii).and.e.ge.egp(ii+1)) then
          ig=ii
          go to 1180
        endif
1178  continue
c
c dunn
c
c*****fill the fission bank
 1180 call  move(fsbank(1,ifl),x,lnfbnk)
      if (nesta) call  move(fsbank(lfstka,ifl),stacka,lnstka)
```

```
      if (nesth) call  move(fsbank(lfstkh,ifl),stackh,lnstkh)
 1190 fisw   = fisw-rakbar
      if (fisw.gt.0.0) go to 1130
      lfisflg=0
c*********************************end fission ******************************
 1200 kcol   = ki
      if ( lab ) go to 1220
      ig     = ignew
c
c  dunn
c  modification:  added new energy after collision
      e      = enew
c  dunn
c
      nsig   = isggc(ig)
      newsg  = lsg(ig)
      if (nextsg.eq.newsg) go to 140
c*****transfered to a different supergroup
 1210 if (ltrk) call  trkwrt('newsg   ',nsig,nextsg)
      ninf   = ninf-1
      if (lnmhol) iextra(1) = nhole
      if (lmult) extra(iself) = fself
      call  move(nubank(1,ninf),wt,lnubnk)
      if (nesta) call  move(nubank(lstka,ninf),stacka,lnstka)
      if (nesth) call  move(nubank(lstkh,ninf),stackh,lnstkh)
      neucnt(newsg) = neucnt(newsg)+1
      go to 100
 1220 continue
      if (ltrk) call  trkwrt('death   ',nsig,nextsg)
      tlf    = tlf + wt*tme
      go to 100
10000 format('0keno message number k5-128',10x,
     * ' neutron bank is full.  splitting not allowed')
10100 format('0keno error number k5-62',10x,
     * ' posit error in track --- illegal geometry type'/' x =',1pe12.5,
     * ' y =',e12.5,' z =',e12.5,' k1 =',i5,' k2 =',i5,' k =',i5,
     * ' igeo =',i5)
      end
```

Appendix B

Listing of PKENO V.a Subroutine COLLISION

```
      subroutine collision(lfstart,iregc,flux,pth,igold,rpth,lab,ic,
     * ab,fisw,fap,fnfp,fnap,fisn,g,lself,impc,imp,matrix,
     * tp,kft,tu,lls,larray,nar,irega,iextra,ta,lhole,
     * nhl,nhole,stackh,th,wtavg,ninf,neut,nscnt,lnmhol,
     * nubank,ignew,nsct1,lreturn,lglobl,
     * mat,vinv,lensg,stacka,
     * npmix,nxss,jxss,xss,dens,numix,nm,nsc,nlg,sigt,xsd,enew,
     * sig,inmt,egp,ith,chi,lenchi,lnchi,lfisflg,mt,nsmt,nsab,temp,spl,
     * vnuc,vnr,irt,sin,sel,jic,jec,ei1c,ei2c,ee1c,ee2c)
c
c.....
      common /albnam/rnames(6)
      character*8 rnames
      common /albdat/ intr(6),idalb(6),nbxl(6),iface,nsig,
     * nalb,nang,ng,awt,lprbx,lprba,nagsg,nag1
c.....
c.....
      common /dimen/  tmax,  tbtch,  dwtav, wthigh,  wtlow,    big,
     1         nba,    npb,  nskip,   nbas, nrstrt, numx1d,
     2       nbank,  nxnbk,  nfbnk,  nxfbk,
     3       lbank,  lfbnk, lnubnk, lnextr, lnfbnk,
     4         ngp,   ngp1, ntypst,   nmat,   matt,   nmix,
     5        kmax,  krefm,   nbox,  nboxt,  ngblu,  nucom,
     6      nbxmax, nbymax, nbzmax, nglobl, matdim,  nacom,
     7      maxmix, maximp,  mang,   nimp, numids,    ncs, ntsets,
     8         nsg, nchigp,  nsct,    mix,   mixt,    npl,   pbxs,
     9      maxara, lbalng,  ndelx,  ndely,  ndelz,
     a      numara, nalvls, lnstka,  lstka, lfstka,    iaa,
     b      numhol, nhlvls, lnstkh,  lstkh, lfstkh,    ihh,
     c      mult
c
c       tmax   is time allowed for the problem
c       tbtch  is time allowed for a generation
c       dwtavg is the default value of avg. wt.
c       wthigh is the wt. at which splitting occurs (wthigh*wtavg)
c       wtlow  is the russian roulette threshold (wtlow*wtavg)
c       big    is the maximum path in a void
c
c       nba    is number of generations
c       npb    is number per generation
c       nskip  is number of generations skipped
c       nbas   is the starting generation
c       nrstrt is generations between restarts
c       numx1d is no. of extra 1-d xsecs
c       nbank  is no. of neutrons that can be stored in neutron bank
c       nxnbk  is extra positions per neutron in neutron bank
c       nfbnk  is no. of neutrons that can be stored in fission bank
c       nxfbk  is extra positions per neutron in fission bank
c
c       lbank  is total positions per neutron in the neutron bank
c       lfbank is total positions per neutron in the fission bank
```

```
c       lnubnk is the length of common/nutron/ in the neutron bank
c       lnextr is length of extra data from subroutine extra
c       lnfbnk is the length of common/nutron/ in the fission bank
c       ngp    is number of energy groups
c       ngp1   is no. of energy groups + 1
c       ntypst is start type
c       nmat   is number of mixtures on ice tape
c       matt   is no. of mixtures in the problem
c       nmix   is no. of mixing table entries
c       kmax   is number of geometry regions used
c       krefm  is no. of geometry cards read
c       nbox   is the largest unit number specified in the input data
c       nboxt  is the largest unit number in the problem
c       ngblu  is the global unit number
c       nucom  is the number of units having comments in the geometry
c       nbxmax is number of units in the x dir.
c       nbymax is number of units in the y dir
c       nbzmax is number of units in the z direction
c       nglobl is the global array number
c       maxmix is largest mixture no. in geometry
c       maximp is largest biasing region no. in geometry
c       mang   is no. of angles on xsec tape
c       nimp   is no. of biasing regions
c       numids is the number of bias id's requested
c       ncs    is the number of sets of weights read from cards
c       ntsets is the total number of weight group structures read
c                from cards
c       matdim is nbxmax*nbymax*nbzmax
c       nacom  is the number of arrays with comments in the array data
c       nsg    is the number of supergroups
c       nchigp is the no. of groups the fission spectrum varies with
c       nsct   is the number of scattering angles
c       mix    is the no. of different mixtures to be mixed
c       mixt   is the maximum mixture no. to be mixed
c       npl    is the order of legendre coefficients + 1
c       pbxs   is the threshold for printing messages for errors found
c                in processing the angles and probabilities
c       maxara is the maximum array number
c       lbalng is the length of the composite lba array
c       ndelx  is the length of the composite delx array
c       ndely  is the length of the composite dely array
c       ndelz  is the length of the composite delz array
c       numara is the number of arrays
c       nalvls is the max. depth of nesting for arrays
c       lnstka is the length of stacka (array stack)
c       lstka  is the pointer to stacka within the neutron bank
c       lfstka is the pointer to stacka within the fission bank
c       iaa    is the location in the fission bank of the pointer to
c                last level in the stacka array iextra(ias)
c       numhol is the number of holes
c       nhlvls is the max. depth of nesting for holes
```

```
c       lnstkh is the length of stackh (hole stack)
c       lstkh  is the pointer to stackh within the neutron bank
c       lfstkh is the pointer to stackh within the fission bank
c       ihh    is the location in the fission bank of the pointer to
c                last level in the stackh array iextra(ihs)
c       mult   is the variable setting the number of storage locations
c                used by a variable typed double precision
c.....
c.....
      common/final/dtime,diin,diii,igen,iostrt
c.....
c.....
      common /lifetm/ sorwt,tofis,fleakt,sleak,sqleak,fmabst,smabs,
     1 sqabs,fmfist,sfmfis,sqfis,tfisn,gfis,nubar,nusqr,
     2 timg,tmgs,timl,tmls,sqkef,skbar,akbar,rakbar,age,tlf,gbar,gsqr,
     3 self,slfsq,tself,awtsum,awtsq,lvel,lprnt,indxfs
      double precision sorwt,tofis,fleakt,sleak,sqleak,fmabst,smabs,
     1 sqabs,fmfist,sfmfis,sqfis,tfisn,gfis,nubar,nusqr,
     2 timg,tmgs,timl,tmls,sqkef,skbar,akbar,rakbar,age,tlf,gbar,gsqr,
     3 self,slfsq,tself,awtsum,awtsq
c.....gbar   = sum of the average energy group where fission occurs
c.....gsqr   = sum of the squares of the group where fission occurs
c.....nubar  = sum of the average value of nu
c.....nusqr  = sum of the squares of nu
c.....self   = avg. self multiplication of the units
c.....slfsq  = sum of the squares of the self multiplication
c.....indxfs = position of the fission xsec in the extra 1-d xsec array
c.....tfisn  = avg. fission cross section
c.....gfis   = fission weighted energy group
c.....
c.....
      common /logic/  lrun,  lplot,  nflx,  nfden,  lmult, newbar,
     1       lunit,  lcku, pmunit, larpos,  lckp, pmapos,
     2      lmhole,  lckh, pmhole, lhhgh,
     3      lmarry,  lcka, pmarry, lahgh,
     4       lpaxs,  prt1, prtp0, prtap, prtchi, prtex,
     5         lfa, lcorsp,  lpwt, lgeom, ldbug,  ltrk,  nadj,
     6      lstgen, lxtra, lsgun,  lsav, lstop,  mflag, lreed,
     7        lnxx, lsavf,  lrnd, lsavu, lsavp,  lsavh, lsava,
     8        mbox, exrfl,  nesta, nesth, ppmsg
      logical nflx,nfden,nadj,lpaxs,prtap,prt1,prtp0
      logical lfa,lunit,larpos,lmhole,lmarry,lsgun,lsav
      logical pmapos,pmunit,pmhole,pmarry
      logical lstop,mflag,lreed
      logical prtchi, prtex
      logical ldbug,ltrk
      logical lnxx,lahgh,lhhgh
      logical lsavf,lrnd,lsavu,lsavp,lsavh,lsava,mbox,exrfl
      logical lpwt,lgeom
      logical lmult,newbar,nesta,nesth,lrun,lplot,ppmsg
      logical lcorsp,lcku,lckp,lckh,lcka,lxtra(2),lstgen
```

```
c.....lrun is the execution flag.  a value of false suppresses
c     execution.  default is true.
c.....nadj is the adjoint flag.  default is false.
c.....nflx is the flag for collecting and printing fluxes.
c     default is false.
c.....nfden is the flag for collecting and printing fission densities.
c     default is false.
c.....lfa is the flag for printing fissions and absorptions by region.
c     default is false.
c.....lmult is true if the average self multiplication of a unit is to
c     be calculated.
c.....newbar is true if nu-bar and the average fission group are to be
c     calculated
c.....lunit is the logical key for collecting matrix information by
c     unit type.
c.....lcku is the logical flag for calculating cofactor k-effective
c     by unit type
c.....pmunit is the flag for printing the fission probability matrix by
c     unit type.
c.....larpos is the logical key for collecting matrix information by
c     array position  (location within the array).
c.....lckp is the logical flag for calculating cofactor k-effective by
c     unit location or position in the array.
c.....pmapos is the flag for printing the fission probability matrix by
c     unit location or position in the array.
c.....lmhole is the logical key for collecting matrix information by
c     hole.
c.....lckh is the logical flag for calculating cofactor k-effective by
c     hole.
c.....pmhole is the flag for printing the fission probability matrix by
c     hole.
c.....lmarry is the logical key for collecting matrix information by
c     array.
c.....lcka is the logical flag for calculating cofactor k-effective by
c     array.
c.....pmarry is the flag for printing the fission probability matrix by
c     array.
c.....lpaxs is the print flag to turn on all mixture cross section
c     print options.  if any options are entered in addition to lpaxs,
c     they override lpaxs
c.....prt1 is the print flag for the 1-d cross sections
c.....prtp0 is the print flag for the 2-d cross sections (p0)
c.....prtap is the print flag for the angles and probabilities
c.....prtchi is the print flag for chi, the fission spectrum
c.....prtex is the print flag for the extra 1-d cross sections
c.....lcorsp is true for printing xsec-albedo correspondence arrays
c.....lpwt is logical key for printing weight average
c.....ldbug keys input debug print
c.....ltrk keys debug tracking print
c.....lgeom is print flag for input geometry data in kenog & rdbox
c.....lplot is the logical flag indicating whether a picture plot
```

```
c     is to be generated.  defalult value is false.
c
c.....lahgh is true if matrix information by array is collected
c     at the highest level and false if at the lowest level.
c.....lhhgh is true if matrix information by hole is collected
c     at the highest level and false if at the lowest level.
c.....lxtra holds space for logical input parameters
c.....lstgen is set true upon completion of the last generation that
c     is run.  it is used for restart in the search routines.
c.....lsgun is set true for a single unit problem.
c.....lsav is false if fissions and absorptions by region were written
c     on restart and rstrt=wstrt and the restarted problem says to
c     suppress fissions and absorptions by region.  a warning message
c     is written, fissions and absorptions by region are calculated
c     but not printed
c.....lstop is used to bypass data reading in datain if an end data was
c     encountered before going to datain.
c.....mflag is set true when an error is found, so execution can be
c     terminated when data processing is completed.
c.....lreed is true if a read flag was read when the end param flag
c     was expected.
c.....lnxx is true if an albedo option is to be exercised
c.....lsavf is false if fluxes were written on restart and rstrt=wstrt
c     and the restarted problem says to suppress the fluxes.  a warning
c     message is written and the fluxes are calculated but not printed.
c.....lrnd is true if the restarted problem is to continue with the
c     original random sequence.  to change random sequence, enter a
c     random number as a parameter
c.....lsavu is false if matrix information by unit type was written
c     on the restart unit and rstrt=wstrt and the restarted problem
c     specifies suppressing it.  a warning message is printed and
c     the data is calculated but not printed.
c.....lsavp is false if matrix information by unit position was
c     written on the restart unit and rstrt=wstrt and the restarted
c     problem specifies suppressing it.  a warning message is printed
c     and the data is calculated but not printed.
c.....lsavh is false if matrix information by hole was
c     written on the restart unit and rstrt=wstrt and the restarted
c     problem specifies suppressing it.  a warning message is printed
c     and the data is calculated but not printed.
c.....lsava is false if matrix information by array was
c     written on the restart unit and rstrt=wstrt and the restarted
c     problem specifies suppressing it.  a warning message is printed
c     and the data is calculated but not printed.
c.....mbox is true if multiple units are specified
c.....exrfl is true if a global reflector is present
c.....nesta is logical flag for nested arrays. default=false
c.....nesth is logical flag for nested holes.  default=false
c.....ppmsg is the flag for printing posit error messages.
c     ppmsg is false while drawing a picture and true while tracking.
c.....
```

```
c.....
      common /lowbnd/ l01,l02,l03,l04,l05,l06,l07,l08,l09,l10,
     *               l11,l12,l13,l14,l15,l16,l17,l18,l19,l20
c     l01 is the lower bound for numhol
c     l02 is the lower bound for nalb
c     l03 is the lower bound for nang
c     l04 is the lower bound for lnextr
c     l05 is the lower bound for lbalng
c     l06 is the lower bound for numara and maxara
c     l07 is the lower bound for nalvls
c     l08 is the lower bound for nhlvls
c     l09 is the lower bound for numx1d
c     l10 is the lower bound for nsct1
c     l11 is the lower bound for nsct
c     l12 is the lower bound for ndelx,ndely,ndelz
c     l13 is the lower bound for lprbx
c     l14 is the lower bound for lprba
c.....
c.....
      common /matrx/ nbxgbl,nbygbl,nbzgbl,kglobl,kgore,ngx,ngy,ngz,ng01
c     nbxgbl,nbygbl,nbzgbl are nbxmax,nbymax,nbzmax in the global array
c     kglobl is kcore of the global array
c     ngx,ngy,ngz define the current position in the global array
c     kgore is kcore of the first array level at that position
c     ng01 is the lower index on the global lba
c.....
c.....
      common/nutron/    wt,    tme,     u,     v,     w,
     1        nunum,   kcol,     x,     y,     z,
     2        kcore,    nbx,    nby,    nbz,     k,    ig,  e, nutn,
     3    extra(10),     kr,     ki,     ll,    k1,    k2,  igeo,
     4           x1,     y1,     z1,   xxk1,   xxk2,   xxk3,
     5         xxk4,   xxk5,   xxk6,   xxk7,   xxk8,
     6            m,      n, etausd, nsplit,  nfiss,
     7        iself,    ias,    ihs,    now,
     8       imatr1, imatr2, imatr3, imatr4
      dimension iextra(10)
c     equivalence (extra(1),iextra(1))
c.....
c.....
      common /titl/ title(20),rdwrd
      character*4 title,rdwrd
c     title is the problem title
c     rdwrd is the block identifier specified on the first data block
c.....
c.....
      common /unit/ inpt, outpt, icexs, albdo, wts, skrt, rstrt, wstrt,
     1 ampxs, direct(5), i0, i1, i2, i3, i4, nspare(4),igs
      integer outpt,albdo,skrt,rstrt,wstrt,direct,wts,icexs,ampxs
c.....
c.....
```

```
      common /split/ ithn,sigtith,sige,sigins,sigam,siga,signp,sigf,
     1  ltherm,ji,je,ei1,ei2,ee1,ee2,nsi,leel,lein,neel,nein,selith,
     2  sinith,neutn
c.....
c.....
c******fsbank contains x,y,z,nbx,nby,nbz,k,energy gp (nsig) ************
      dimension mat(krefm),
     *          nubank(lbank,nbank)
      dimension wtavg(lensg,nimp),vinv(lensg),
     *          impc(maximp),imp(krefm),
     *          flux(lensg,kmax)
      dimension tp(matdim,matdim),tu(nboxt,nboxt)
      dimension th(l01:numhol,l01:numhol),ta(l06:numara,l06:numara)
      dimension iregc(krefm)
      dimension irega(l06:maxara)
c     ndxlba holds the starting location of each lba array
c     irega is the array correspondence array
c     indxx,indxy,indxz are offsets into deltx,delty,deltz
c     nbxmx,nbymx,nbzmx contain nbxmax,nbymax,nbzmax for each array
c     sold is the beginning point of the track
c     snew is the crossing point
c     s1old is the end point of the track
c     holx, holy, holz are the x,y&z offset of the hole
c     lholu is the unit within the hole
c     khole is the region that contains the hole
c     ifh and ilh are the first and last holes in the region
      dimension stacka(4,l07:nalvls),stackh(l08:nhlvls)
c
      dimension nxss(16,nm+nsc),jxss(32,nm+nsc),xss(nlg,nm+nsc),
     *dens(nm),npmix(2*numix),sigt(nm),xsd(nm+nsc,9),sig(66),
     *inmt(66),egp(ngp1),chi(lnchi),lenchi(nm),nsmt(nsc),
     *nsab(nm),temp(nm),spl(23,nbank),vnuc(3),vnr(3),sin(nm),sel(nm)
c
      integer stacka,stackh
      dimension uvw(3),nxyz(3),ih(6),xxedge(3),irev(6),logflg(6)
      equivalence (u,uvw(1)),(nbx,nxyz(1)),(x,xxedge(1))
      equivalence (logxp,logflg(1)),(logxm,logflg(2)),(logyp,logflg(3)),
     *           (logym,logflg(4)),(logzp,logflg(5)),(logzm,logflg(6))
      logical logflg
      logical larray,ic,lab,logxp,logxm,logyp,logym,logzp,logzm,
     *        lhole,lnmhol
c     lrghol logical flag is true if the region contains holes
c     lhole  logical flag is true if you are in a hole
c     lnmhol logical flag is true if the problem utilizes holes
c      equivalence(fself,lself)
      logical lself, lglobl
      logical matrix
      double precision fleak, fmabs, fmfis, psi, toosml
      double precision fltrn, sflra, exprn
      double precision summtd, sigind
      parameter ( toosml = 1.0d-10 )
```

```
      data ih/1,1,2,2,3,3/
      data irev/2,1,4,3,6,5/
c
c
c  if lfstart=1, the proceed to splitting and russian
c  roulette
      if (lfstart.eq.1) go to 3000
c
c*******************************start collision*****************************
  930 x    = x1
      y    = y1
      z    = z1
      if (nflx) flux(nsig,iregc(k)) = flux(nsig,iregc(k)) + wt*pth
c
      tme   = tme + pth*vinv(nsig)
      igold = ig
      rpth  = 0.0
      lab   = .false.
      ic    = .true.
c
      ith  = 0
      isum = 0
      ab   = 0.
      fisw = 0.
      fnap = 0.
      fap  = 0.
      fnfp = 0.
      wtnu = 0.
      le   = 0
      ne   = 0
      nr   = 0
      eout = 0.
      enew = 0.
      er   = 0.
      ignew = 0.
      elab = 0.
      um   = 0.
      ucom = 0.
      ulab = 0.
      linel = 0
      lcorct = 0
      ltherm = 0
c
      sigs  = 0.
      sige  = 0.
      sigins = 0.
      sigin = 0.
      signp = 0.
      sigf  = 0.
      siga  = 0.
      sigam = 0.
```

```
c
      pel       = 0.
      pin       = 0.
      pinel     = 0.
      lscatter  = 0.
      ltscatter = 0.
c
c     call clear (sigt,nm)
      call clear (sig,66)
      call clear (inmt,66)
c
c************************select the collision nuclide**********************
c
      isum  = npmix(kr+numix)
      l     = 2
      if (npmix(kr).eq.1) then
         ith=isum+1
         sigt(ith) = sigt(ith)
         go to 955
      endif
c     if (npmix(kr).eq.1) then
c        ith=isum+1
c        le=jxss(1,ith)
c        ne=nxss(3,ith)
c        call locat(xss(le,ith),ne,e,j)
c        if (j.eq.0.or.j.eq.ne)then
c              if (j.eq.0)sigt(ith)=xss(le+ne,ith)
c              if (j.eq.ne)sigt(ith)=xss(le+2*ne-1,ith)
c        else
c              e1=xss(le+j-1,ith)
c              e2=xss(le+j,ith)
c              if(e.lt.e1.or.e.gt.e2) then
c                 linterp=1
c              endif
c              st1=xss(le+j-1+ne,ith)
c              st2=xss(le+j+ne,ith)
c              if(e.eq.e1)st=st1
c              if(e.eq.e2)st=st2
c              if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
c     *           st1,e2,st2,st)
c              sigt(ith)=st
c        endif
c        go to 955
c     endif
c  if more than one isotope in mixture, randomly select the isotope
c     do 935 ii=1,npmix(kr)
c           ith=isum+ii
c           le=jxss(1,ith)
c           ne=nxss(3,ith)
c           call locat(xss(le,ith),ne,e,j)
c           if (j.eq.0.or.j.eq.ne)then
```

```
c               if (j.eq.0)sigt(ii)=dens(ith)*xss(le+ne,ith)
c               if (j.eq.ne)sigt(ii)=dens(ith)*xss(le+2*ne-1,ith)
c             else
c               e1=xss(le+j-1,ith)
c               e2=xss(le+j,ith)
c               if(e.lt.e1.or.e.gt.e2) then
c                  linterp=1
c               endif
c               st1=xss(le+j-1+ne,ith)
c               st2=xss(le+j+ne,ith)
c               if(e.eq.e1)st=st1
c               if(e.eq.e2)st=st2
c               if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
c     *            st1,e2,st2,st)
c               sigt(ii)=dens(ith)*st
c             endif
c935   continue
      do 935 ii=1,npmix(kr)
            ith=isum+ii
            sigt(ith)=dens(ith)*sigt(ith)
935   continue
c  randomly select the nuclide
      sum=0.
      do 938 ii=1,npmix(kr)
            ith = isum+ii
            sum=sum+sigt(ith)
938   continue
            r = fltrn()
            sum=r*sum
      do 950 kk=1,npmix(kr)
            sum1=0.
            do 940 jj=1,kk-1
                  sum1=sum1+sigt(isum+jj)
940          continue
            sum2=0.
            do 945 jj=1,kk
                  sum2=sum2+sigt(isum+jj)
945          continue
            if (sum1.lt.sum.and.sum2.ge.sum) then
            ith=isum+kk
            go to 955
            endif
950   continue
c*******************calculate probabilities*******************************
955   if (npmix(kr).gt.1)sigt(ith)=sigt(ith)/dens(ith)
c note:  sigam is the absorption cross section specifically for mcnp.
c        it only includes neutron disappearance reactions---it does
c        not include fission, (n,2n), (n,3n), etc.
      a = xsd(ith,1)
      le=jxss(1,ith)
      ne=nxss(3,ith)
```

```
      call locat(xss(le,ith),ne,e,j)
      if (j.eq.0.or.j.eq.ne)then
              if (j.eq.0)sigam=xss(le+2*ne,ith)
              if (j.eq.ne)sigam=xss(le+3*ne-1,ith)
      else
               e1=xss(le+j-1,ith)
               e2=xss(le+j,ith)
               if(e.lt.e1.or.e.gt.e2) then
                  linterp=1
               endif
               sa1=xss(le+j-1+2*ne,ith)
               sa2=xss(le+j+2*ne,ith)
               if(e.eq.e1)sa=sa1
               if(e.eq.e2)sa=sa2
               l=2
               if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
     *            sa1,e2,sa2,sa)
               sigam=sa
      endif
c*****determine if s(a,b) thermal data should be used to model the collision
      if (nsab(ith).ne.0) then
        if (sin(ith).ne.0..or.sel(ith).ne.0.) then
          ji    = jic
          je    = jec
          ei1   = ei1c
          ei2   = ei2c
          ee1   = ee1c
          ee2   = ee2c
          nsi   = nsab(ith)
          lein  = jxss(1,nsi)
          leel  = jxss(4,nsi)
          nein  = xss(lein,nsi)
          neel  = xss(leel,nsi)
          ltherm = 1
          sigs = sel(ith) + sin(ith)
          siga = sigam
          sigf = 0.
          vbar = 0.
          fap  = siga/sigt(ith)
          fnap = sigs/sigt(ith)
          fnfp = 0.
          go to 2000
        end if
      endif
c*****non s(a,b) cross sections*****************************************
c*****elastic cross section
1400  le=jxss(1,ith)
      ne=nxss(3,ith)
      call locat(xss(le,ith),ne,e,j)
      if (j.eq.0.or.j.eq.ne)then
              if (j.eq.0)sige=xss(le+3*ne,ith)
```

```
            if (j.eq.ne)sige=xss(le+4*ne-1,ith)
      else
            e1=xss(le+j-1,ith)
            e2=xss(le+j,ith)
            if(e.lt.e1.or.e.gt.e2) then
               linterp=1
            endif
            se1=xss(le+j-1+3*ne,ith)
            se2=xss(le+j+3*ne,ith)
            if(e.eq.e1)se=se1
            if(e.eq.e2)se=se2
            l=2
            if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
     *         se1,e2,se2,se)
            sige=se
      endif
c     if (nxss(5,ith).eq.0) then
c        sigs   = sige
c        sigins = 0.
c        signp  = 0.
c        siga   = sigam
c        fap    = siga/sigt(ith)
c        fnap   = sigs/sigt(ith)
c        fnfp   = 0.
c        vbar   = 0.
c        sigf   = 0.
c        go to 2000
c     endif
c*****inelastic scattering cross section
      sumins = 0.
      sumnp  = 0.
      do 1500 ii=1,nxss(4,ith)
        mt = xss(jxss(3,ith)+ii-1,ith)
        if (mt.ge.51.and.mt.le.91) then
          lxs      = jxss(6,ith)
          locai    = xss(lxs+ii-1,ith)
          if (locai.eq.0) then
             locai=locai
          endif
          iei      = xss(jxss(7,ith)+locai-1,ith)
          nei      = xss(jxss(7,ith)+locai,ith)
          lcrs     = jxss(7,ith)+locai+1
          call locat(xss(le+iei-1,ith),nei,e,j)
          if (j.eq.0.or.j.eq.nei)then
             if (j.eq.0)sigins=xss(lcrs,ith)
             if (j.eq.nei)sigins=xss(lcrs+nei-1,ith)
          else
             e1=xss(le+iei-1+j-1,ith)
             e2=xss(le+iei-1+j,ith)
             if (e.lt.e1.or.e.gt.e2) then
                e = e
```

```
            endif
            sig1=xss(lcrs-1+j,ith)
            sig2=xss(lcrs-1+j+1,ith)
            if(e.eq.e1)sigins=sig1
            if(e.eq.e2)sigins=sig2
            l=2
            if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
     *         sig1,e2,sig2,sigins)
          endif
          sumins = sumins + sigins
        else
          if (mt.lt.50.and.mt.ne.18.and.mt.ne.10.and.mt.ne.4.and.
     *       mt.ne.5.and.mt.ne.27.and.mt.ne.31.and.mt.ne.39.and.
     *       mt.ne.40) then
            lxs      = jxss(6,ith)
            locai    = xss(lxs+ii-1,ith)
            iei      = xss(jxss(7,ith)+locai-1,ith)
            nei      = xss(jxss(7,ith)+locai,ith)
            lcrs     = jxss(7,ith)+locai+1
            call locat(xss(le+iei-1,ith),nei,e,j)
            if (j.eq.0.or.j.eq.nei)then
              if (j.eq.0)signp=xss(lcrs,ith)
              if (j.eq.nei)signp=xss(lcrs+nei-1,ith)
            else
              e1=xss(le+iei-1+j-1,ith)
              e2=xss(le+iei-1+j,ith)
              sig1=xss(lcrs-1+j,ith)
              sig2=xss(lcrs-1+j+1,ith)
              if(e.eq.e1)signp=sig1
              if(e.eq.e2)signp=sig2
              l=2
              if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
     *           sig1,e2,sig2,signp)
            endif
            sumnp = sumnp + signp
          endif
        endif
1500  continue
            sigins = sumins
            signp  = sumnp
            if (sigins.lt.0.)then
                 sigins = sigins
            endif
            sigs   =  sige  + sigins
            siga   =  signp + sigam
            fap    =  siga/sigt(ith)
            fnap   =  sigs/sigt(ith)
c            if (jxss(2,ith).eq.0) then
c              fnfp = 0.
c              vbar = 0.
c              sigf = 0.
```

```
c                  go to 2000
c               endif
c*********************determine fission probability fnfp*****************
c
c******read nu data
c nu is provided in either polynomial or tabular form
      le2=jxss(2,ith)
      lnub=xss(le2,ith)
      if (le2.eq.0) then
        fnfp = 0.
        vbar = 0.
        sigf = 0.
        go to 2000
      endif
      if (lnub.lt.0) then
        knu=le2 + abs(lnub) + 1
      else
        knu=le2
      endif
      lnu=xss(knu,ith)
      go to (1600,1700) lnu
c
c******functional form of nu
1600  nc=xss(knu+1,ith)
      sumnub = 0.
      do 1610 ii=1,nc
        c     = xss(knu+1+ii,ith)
        vbar = c*(e**(ii-1))
        sumnub = sumnub + vbar
1610  continue
      vbar  =  sumnub
      go to 1800
c
c******tabular form of nu
1700  nr=xss(knu+1,ith)
      ne=xss(knu+2+2*nr,ith)
      call locat(xss(knu+3+2*nr,ith),ne,e,j)
      if (j.eq.0.or.j.eq.ne)then
              if (j.eq.0)vbar=xss(knu+3+2*nr+ne,ith)
              if (j.eq.ne)vbar=xss(knu+3+2*nr+ne+ne-1,ith)
      else
               e1=xss(knu+3+2*nr+j-1,ith)
               e2=xss(knu+3+2*nr+j,ith)
               if(e.lt.e1.or.e.gt.e2) then
                  linterp=1
               endif
               vbar1=xss(knu+3+2*nr+j-1+ne,ith)
               vbar2=xss(knu+3+2*nr+j+ne,ith)
               if(e.eq.e1) vbar=vbar1
               if(e.eq.e2) vbar=vbar2
               l=2
```

```
            if (nr.gt.0) then
               jj=j
               call locnbt (xss(knu+2,ith),nr,jj,j)
               l=xss(knu+2+nr+j-1,ith)
            endif
            if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
     *         vbar1,e2,vbar2,vbar)
      endif
 1800 continue
c
c******read fission cross sections
c
      nmt = nxss(4,ith)
      lmt = jxss(3,ith)
      lxs = jxss(6,ith)
      sumf= 0.
      do 1900 ii=1,nmt
        mt  = xss(lmt+ii-1,ith)
        if (mt.eq.18.or.mt.eq.19.or.mt.eq.20.or.mt.eq.21.
     *      or.mt.eq.38) then
          locai = xss(lxs+ii-1,ith)
          if (locai.eq.0) then
            locai=locai
          endif
          iei   = xss(jxss(7,ith)+locai-1,ith)
          nei   = xss(jxss(7,ith)+locai,ith)
          lcrs  = jxss(7,ith)+locai+1
          call locat(xss(le+iei-1,ith),nei,e,j)
          if (j.eq.0.or.j.eq.nei)then
             if (j.eq.0)sigf=xss(lcrs,ith)
             if (j.eq.nei)sigf=xss(lcrs+nei-1,ith)
          else
              e1=xss(le+iei-1+j-1,ith)
              e2=xss(le+iei-1+j,ith)
              if(e.lt.e1.or.e.gt.e2) then
                 linterp=1
              endif
              sigf1=xss(lcrs-1+j,ith)
              sigf2=xss(lcrs-1+j+1,ith)
              if(e.eq.e1)sigf=sigf1
              if(e.eq.e2)sigf=sigf2
              l=2
              if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
     *           sigf1,e2,sigf2,sigf)
              sigf=sigf
          endif
          sumf = sumf + sigf
          if (mt.eq.18) go to 1910
        endif
 1900 continue
 1910 sigf = sumf
```

```
      fnfp = (vbar*sigf)/sigt(ith)
c**********************calculate weights*********************************
2000  ab    = wt*fap
      fisw  = wt*fnfp
      wtnu  = wt*fnap
c*****************************************************************************
      if (newbar) then
        fisn   = wt*sigf/sigt(ith)
        g      = ig*fisn
      end if
      if (lself) tself  = tself+fisw
      wt     = wtnu
      ki     = impc(imp(k))
      if (ltrk) call  trkwrt('collison',nsig,nextsg)
      if (matrix) then
        kftf   = (ngz-1)*nbxgbl*nbygbl+(ngy-1)*nbxgbl+ngx
        if (.not.lglobl)  kftf   = matdim
        if (larpos) tp(kft,kftf) = tp(kft,kftf)+fisw
        if (lunit)  tu(lls,ll)   = tu(lls,ll)+fisw
        if (larray .and. lmarry .and. nar.gt.0) then
          narf   = irega(now)
          if (lahgh.and.iextra(ias).gt.0)
     *     narf  = irega(mat(stacka(1,1)))
          ta(nar,narf) = ta(nar,narf)+fisw
        end if
        if (lmhole .and. lhole .and. nhl.gt.0) then
          nhl1   = nhole
          if (lhhgh.and.iextra(ihs).gt.0) nhl1  = stackh(1)
          th(nhl,nhl1) = th(nhl,nhl1)+fisw
        end if
      end if
c*******check to see if the neutron should be split**********************
c
c  modification:  changed line 960 in TRACK to 3000
3000  continue
c
      if (wt.gt.wthigh*wtavg(nsig,ki)) then
c****** check to see if the bank can hold a split neutron **************
        if ((ninf-neut).lt.3) then
          mflag  = nscnt.gt.50
          write(outpt,10000)
c
c modification: the following return statement should be a return out
c               of track.  therefore, set the flag lreturn=1. when
c               control is returned to track, the flag will indicate
c               a return back to guide
c         if (nscnt.gt.50) return
          if (nscnt.gt.50) then
            lreturn=1
            return
c
```

```
        endif
        nscnt  = nscnt+1
      else
c******** split the neutron and bank it **********************************
        if (lfstart.ne.1) then
          ithn   = ith
          nutn   = neut
          neutn  = nutn
          sigtith = sigt(ith)
          selith  = sel(ith)
          sinith  = sin(ith)
          call move(spl(1,neut),ithn,23)
        endif
        wt     = wt/2.0
        neut   = neut+1
        nunum  = -nunum
        if (lnmhol) iextra(1) = nhole
        if (ltrk) call  trkwrt('split   ',nsig,nextsg)
c*********call move to store the split neutron*****************************
        call  move(nubank(1,neut),wt,lnubnk)
        if (lfstart.ne.1) call  move(spl(1,neut),ithn,23)
        if (lfstart.eq.1) then
           call  move(spl(1,neut),spl(1,neut-1),23)
           call  move(ithn,spl(1,neut),23)
        endif
        nunum  = -nunum
        if (nesta) call  move(nubank(lstka,neut),stacka,lnstka)
        if (nesth) call  move(nubank(lstkh,neut),stackh,lnstkh)
c
c  modification: changed line from 960 to 3000
c          go to 960
           go to 3000
c
      end if
    end if
c********** play russian roulette*****************************************
    if (wt.lt.wtlow*wtavg(nsig,ki)) then
      lab    = fltrn()*wtavg(nsig,ki).gt.wt
c*****  lab=true indicates the particle will die
      if ( lab ) go to 4600
      wt     = wtavg(nsig,ki)
    end if
c**********load split neutron data if needed*********************************
    if (lfstart.eq.1) then
      call move(ithn,spl(1,neut),23)
      ith       = ithn
      sigt(ith) = sigtith
      sel(ith)  = selith
      sin(ith)  = sinith
    endif
c**********use s(a,b) data if appropriate************************************
```

```
      if (ltherm.eq.1) go to 3400
c*********sample velocity of nuclide if appropriate**********************
      if (ltherm.eq.0.and.(a.le.1.
     *    or.e.lt.3.45e-8*temp(ith))) then
          eold  = e
          lsmpl = 1
          call sampvel(a,temp(ith),e,er,vnuc,vnr,snr,irt,u,v,w)
          e     = er
c          le    = jxss(1,ith)
c          call reread (e,sigt(ith),sige,sigins,signp,sigf,xss(le,ith),
c     *                 jxss(1,ith),nxss(1,ith),nlg)
      endif
c**********determine if collision is elastic or nonelastic***************
3050  if ((sigins+signp).eq.0.) go to 3105
      if (linel.eq.1) then
          summt = 0.
          incnt = 0
          iii   = 0
          do 3100 ii=1,nxss(4,ith)
            mt = xss(jxss(3,ith)+ii-1,ith)
          if (mt.lt.18.and.mt.ne.10.and.mt.ne.4.and.mt.ne.5.or.
     *       mt.gt.21.and.mt.ne.38.and.mt.ne.27.and.mt.ne.31.and.
     *       mt.ne.39.and.mt.ne.40.and.mt.ne.50.and.mt.lt.101) then
c           if (mt.lt.101.and.mt.ne.18.and.mt.ne.10.and.mt.ne.4.and.
c     *       mt.ne.5.and.mt.ne.27.and.mt.ne.31.and.mt.ne.39.and.
c     *       mt.ne.40.and.mt.ne.50) then
            iii       = iii + 1
            inmt(iii) = mt
            lxs       = jxss(6,ith)
            locai     = xss(lxs+ii-1,ith)
            if (locai.eq.0) then
              locai=locai
            endif
            iei       = xss(jxss(7,ith)+locai-1,ith)
            nei       = xss(jxss(7,ith)+locai,ith)
            lcrs      = jxss(7,ith)+locai+1
            call locat(xss(le+iei-1,ith),nei,e,j)
            if (j.eq.0.or.j.eq.nei)then
              if (j.eq.0)sig(iii)=xss(lcrs,ith)
              if (j.eq.nei)sig(iii)=xss(lcrs+nei-1,ith)
            else
              e1=xss(le+iei-1+j-1,ith)
              e2=xss(le+iei-1+j,ith)
              if(e.lt.e1.or.e.gt.e2) then
                 linterp=1
              endif
              sig1=xss(lcrs-1+j,ith)
              sig2=xss(lcrs-1+j+1,ith)
              if(e.eq.e1)sig(iii)=sig1
              if(e.eq.e2)sig(iii)=sig2
              l=2
```

```
            if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
     *           sig1,e2,sig2,sig(iii))
          endif
           summt = summt + sig(iii)
           incnt = incnt + 1
         endif
3100     continue
       sigin  = summt
       linel  = 0
       lcorct = 1
     else
c       sigin = sigt(ith)-sige-sigam
c       sigin = sigt(ith)-sige-sigam-sigf
       if(sigf.eq.0.) sigin = sigins + signp
       if(sigf.ne.0.) sigin = sigins + signp - sigf
       if(sigin.lt.0.) then
         sigin = 0.
       endif
     endif
     pel   = sige / (sige+sigin)
     pinel = sigin / (sige+sigin)
     r     = fltrn()
     if (r.gt.0.and.r.le.pel)  lscatter = 1
     if (r.gt.pel.and.r.le.1.) lscatter = 2
c****************************process collision*******************************
     go to (3105,3200) lscatter
c*************elastic collision in center of mass system*****************
c
c3105  locb = xss(jxss(8,ith),ith)
3105  locb = 1
c      if (locb.eq.0) then
c        r    = fltrn()
c        ucom = 2*r - 1
c      else
        land = jxss(9,ith)+locb-1
        ne   = xss(land,ith)
        land = jxss(9,ith)+locb
        call locat(xss(land,ith),ne,e,j)
        e1   = xss(land+j-1,ith)
        e2   = xss(land+j,ith)
               if(e.lt.e1.or.e.gt.e2) then
                  linterp=1
               endif
        if (e.eq.e1.or.e.eq.e2) then
          if (e.eq.e2) j = j+1
          go to 3110
        endif
        pl1  = (e2-e)/(e2-e1)
        pl2  = (e-e1)/(e2-e1)
        r    = fltrn()
        if (r.gt.0..and.r.le.pl1) go to 3110
```

```
      if (r.gt.pl1.and.r.le.(pl1+pl2)) j = j+1
3110    land = land+ne+j-1
      lc   = xss(land,ith)
      if (lc.eq.0) then
        r    = fltrn()
        ucom = 2*r - 1
        go to 3120
      endif
      r    = fltrn()
c      ib   = 31*r+1
c      bi   = 31*r+1
      ib   = 32*r+1
      bi   = 32*r+1
      if(ib.gt.32) then
          ib=ib
      endif
c      bi   = 32.*r
c      ib   = bi
      lcc  = jxss(9,ith)+lc-1
      ui   = xss(lcc-1+ib,ith)
      ui1  = xss(lcc-1+ib+1,ith)
      ucom = ui + (bi-ib)*(ui1-ui)
c      r    = fltrn()
c      ucom = ui + r*(ui1-ui)
c    endif
c    if (lsmpl.eq.1) go to 3130
c*****outgoing energy and angle
3120  if (a.gt.1.) then
      term1 =  1. + a*ucom
      term2 =  sqrt(1.+(a**2.)+2*a*ucom)
      ulab  =  term1 / term2
    else
      ulab = sqrt(.5+.5*ucom)
    endif
    if (a.gt.1.) then
      alpha =  ( (a-1.)/(a+1.) )**2.
      enew  =  0.5*e*( (1.-alpha)*ucom + 1. + alpha )
    else
      enew  = .5*e*(1.+ucom)
    endif
    go to 4500
c*****************************nonelastic collision************************
c*****select the nonelastic collision
c
c  sum all the available nonelastic cross sections for ith nuclide
3200  if (lcorct.eq.0) then
      summt = 0.
      incnt = 0
      iii   = 0
      do 3210 ii=1,nxss(4,ith)
        mt = xss(jxss(3,ith)+ii-1,ith)
```

```
          if (mt.lt.18.and.mt.ne.10.and.mt.ne.4.and.mt.ne.5.or.
     *       mt.gt.21.and.mt.ne.38.and.mt.ne.27.and.mt.ne.31.and.
     *       mt.ne.39.and.mt.ne.40.and.mt.ne.50.and.mt.lt.101) then
c          if (mt.lt.18.and.mt.ne.10.and.mt.ne.4.and.mt.ne.5.or.mt.gt.21.
c     *     and.mt.ne.38.and.mt.ne.27.and.mt.ne.31.and.mt.ne.39.and.mt.ne.
c     *     40.and.mt.ne.50.and.mt.lt.101) then
c          if (mt.lt.101.and.mt.ne.18.and.mt.ne.10.and.mt.ne.4.and.
c     *       mt.ne.5.and.mt.ne.27.and.mt.ne.31.and.mt.ne.39.and.
c     *       mt.ne.40.and.mt.ne.50) then
            iii       = iii + 1
            inmt(iii) = mt
            lxs       = jxss(6,ith)
            locai     = xss(lxs+ii-1,ith)
            if (locai.eq.0) then
              locai=locai
            endif
            iei       = xss(jxss(7,ith)+locai-1,ith)
            nei       = xss(jxss(7,ith)+locai,ith)
            lcrs      = jxss(7,ith)+locai+1
            call locat(xss(le+iei-1,ith),nei,e,j)
            if (j.eq.0.or.j.eq.nei)then
               if (j.eq.0)sig(iii)=xss(lcrs,ith)
               if (j.eq.nei)sig(iii)=xss(lcrs+nei-1,ith)
            else
               e1=xss(le+iei-1+j-1,ith)
               e2=xss(le+iei-1+j,ith)
               if(e.lt.e1.or.e.gt.e2) then
                 linterp=1
               endif
               sig1=xss(lcrs-1+j,ith)
               sig2=xss(lcrs-1+j+1,ith)
               if(e.eq.e1)sig(iii)=sig1
               if(e.eq.e2)sig(iii)=sig2
               l=2
               if (e.ne.e1.and.e.ne.e2) call interp (l,e,e1,
     *           sig1,e2,sig2,sig(iii))
            endif
           summt = summt + sig(iii)
           incnt = incnt + 1
         endif
3210     continue
          summtd = summt
          sigind = sigin
          diff   = (abs(sigind-summtd)/sigind)*100.
        if (diff.gt.0.001) then
          linel = 1
          go to 3050
        endif
      endif
c  randomly select the nonelastic reaction
      lcorct =  0
```

```
      r     = fltrn()
      summt = r*summt
      do 3240 kk=1,incnt
            sum1=0.
            do 3220 jj=1,kk-1
                  sum1=sum1+sig(jj)
3220          continue
            sum2=0.
            do 3230 jj=1,kk
                  sum2=sum2+sig(jj)
3230          continue
            if (sum1.lt.summt.and.sum2.ge.summt) then
              do 3235 ii = 1, nxss(4,ith)
                mt = xss(jxss(3,ith)+ii-1,ith)
                if (inmt(kk).eq.mt) then
                  jth=ii
                  go to 3250
                endif
3235          continue
            endif
3240  continue
c*********read tyr block for reaction information**********************
c  determine the number of secondary neutrons and the system for
c  the angular distribution for these emitted neutron(s)
c  ity = - (com system), = + (lab system)
c
c1250  if (inmt(kk).lt.50.or.inmt(kk).gt.91)
c     *wt   =  wtnu - (sig(kk)/sigt(ith))*wtold
3250  lty  =  jxss(5,ith)-1+jth
      ity  =  xss(lty,ith)
      if(abs(ity).eq.1) yld  =  1.
c      if (ity.eq.19) yld = vbar
      if (ity.ne.19.and.abs(ity).gt.1.and.abs(ity).lt.100)then
           yld = abs(ity)
      endif
      if (abs(ity).gt.100) then
           ity=ity
      endif
      if (yld.gt.1.) then
         wt = yld*wt
      endif
      yld = 1.
3300  locb = xss(jxss(8,ith)+jth,ith)
      if (locb.eq.0) then
        r    = fltrn()
        um   = 2*r - 1
      else
        land = jxss(9,ith)+locb-1
        ne   = xss(land,ith)
        land = jxss(9,ith)+locb
        call locat(xss(land,ith),ne,e,j)
```

```
        e1   = xss(land+j-1,ith)
        e2   = xss(land+j,ith)
               if(e.lt.e1.or.e.gt.e2) then
                  linterp=1
               endif
        if (e.eq.e1.or.e.eq.e2) then
          if (e.eq.e2) j = j+1
          go to 3310
        endif
        pl1  = (e2-e)/(e2-e1)
        pl2  = (e-e1)/(e2-e1)
        r    = fltrn()
        if (r.gt.0..and.r.le.pl1) go to 3310
        if (r.gt.pl1.and.r.le.(pl1+pl2)) j = j+1
3310    land = land+ne+j-1
        lc   = xss(land,ith)
        if (lc.eq.0) then
          r    = fltrn()
          um = 2*r - 1
          go to 3320
        endif
        r    = fltrn()
c       ib   = 31*r+1
c       bi   = 31*r+1
        ib   = 32*r+1
        bi   = 32*r+1
        if (ib.gt.32) then
          ib=ib
        endif
c       bi   = 32.*r
c       ib   = bi
        lcc  = jxss(9,ith)+lc-1
        ui   = xss(lcc-1+ib,ith)
        ui1  = xss(lcc-1+ib+1,ith)
        um = ui + (bi-ib)*(ui1-ui)
c       r    = fltrn()
c       um = ui + r*(ui1-ui)
      endif
c*****sample outgoing energy from appropriate endf law
3320  le     = jxss(1,ith)
      led    = jxss(10,ith)
      locci  = xss(led-1+jth,ith)
      ldis   = jxss(11,ith)
      lengy  = ldis+locci-1
      if (nxss(5,ith).ne.0) then
        call sellaw (xss(le,ith),lengy,locci,ldis,law,llaw,e)
      else
        law=0
      endif
      call endflaw (xss(le,ith),law,llaw,e,eout,ldis,chi,lenchi,
     *              lnchi,egp,ith)
```

```
c*****determine which system angle and energy are sampled
      if (ity.lt.0.0) then
         ucom = um
         ecom = eout
         elab = ecom+(e+2*ucom*(a+1.)*sqrt(e*ecom))/((a+1.)**2)
         ulab = ucom*sqrt(ecom/elab)+(1/(a+1.))*sqrt(e/elab)
         enew = elab
      else
         ulab = um
         enew = eout
c         if (ity.eq.19) then
c            lfisflg = 1
c            enew = e
c         endif
      endif
      go to 4500
c**********************thermal scattering law treatment*******************
c select collision type
3400  if(sel(ith).eq.0.) go to 4000
      if(sin(ith).eq.0.) go to 3500
      pel = sel(ith)/(sel(ith)+sin(ith))
      pin = sin(ith)/(sel(ith)+sin(ith))
      r   = fltrn()
      if (r.gt.0.and.r.le.pel)  ltscatter = 1
      if (r.gt.pel.and.r.le.1.) ltscatter = 2
      go to (3500,4000) ltscatter
c  elastic scattering treatment
3500  enew = e
      nx6  = nxss(6,nsi)
      nx61 = nx6+1
      leu  = jxss(6,nsi)
      if (nx6.eq.-1) then
         r = fltrn()
         ulab = 2*r - 1
         go to 4500
      endif
      go to (4800,4800,3600,3700,3900) nxss(5,nsi)+1
c
c     equally probable angle bins
3600  call stop ('elastic s(a,b) collision specifies equally-probable,
     *     angle bins which are not coded in subroutine collision')
c
c     equally probable discrete angles
3700  r  = fltrn()
      ke = r*nx61+1
      ke = ke-1
      if (je.gt.0.and.je.lt.neel) then
         jn = abs(je-1)*nx61
         u1 = xss(leu+jn+ke,nsi)
         u2 = xss(leu+jn+ke+nx61,nsi)
         if (e.eq.ee1) ulab = u1
```

```
        if (e.eq.ee2) ulab = u2
        if (e.ne.ee1.or.e.ne.ee2) then
c           rho = (ee2-e)/(ee2-ee1)
c           ulab = rho*u1 + (1-rho)*u2
            rho  = (e-ee1)/(ee2-ee1)
            ulab = u1 + rho*(u2-u1)
        endif
      else
        if (je.eq.0) ulab = xss(leu+ke,nsi)
        if (je.eq.neel) ulab = xss(leu+ke+(neel-1)*nx61,nsi)
      endif
      go to 4500
c
c     exact treatment of coherent elastic scattering
3900  call stop ('elastic s(a,b) collision specifies exact,
     *     treatment of coherent elastic scattering which is,
     *     not coded in subroutine collision')
c
c  inelastic scattering treatment
4000  nx4  = nxss(4,nsi)
      nx3  = nxss(3,nsi)
      nx31 = nx3+1
      nx32 = nx3+2
      nx7  = nxss(7,nsi)
      liu  = jxss(3,nsi)
      if (nx7.ne.0) go to 4010
      r  = fltrn()
      ke = r*nx4+1
      go to 4040
4010  r    = fltrn()
      ek   = r*(nx4-3)+1
      if (ek.lt.2.) go to 4020
      ke   = ek + 1
      go to 4040
4020  if (ek.lt.1.5) go to 4030
      ke   = nx4 - 1
      if (ek.lt.1.6) ke = ke + 1
      go to 4040
4030  ke = 2
      if (ek.lt.1.1) ke = 1
4040  kk   = ke-1
      r    = fltrn()
      keu  = r*nx31+1
      go to (4800,4800,4100,4200) nxss(2,nsi)+1
c
c     equally-probable angle bins
4100  call stop ('inelastic s(a,b) collision specifies equally-probable,
     *     angle bins which are not coded in subroutine collision')
c
c     equally-probable discrete angles
4200     if (ji.gt.0.and.ji.lt.nein) then
```

```
          if (ji.eq.1) then
             e01 = xss(liu+kk*nx32,nsi)
             e02 = xss(liu+((nx4-1)*nx32)+(ke*nx32),nsi)
             u01 = xss(liu+kk*nx32+keu,nsi)
             u02 = xss(liu+((nx4-1)*nx32)+(ke*nx32)+keu,nsi)
          else
             e01 = xss(liu+((nx4-1)*nx32)+nx4*(ji-2)*nx32+nx32*ke,
     *            nsi)
             e02 = xss(liu+((nx4-1)*nx32)+nx4*(ji-1)*nx32+nx32*ke,
     *            nsi)
             u01 = xss(liu+((nx4-1)*nx32)+nx4*(ji-2)*nx32+nx32*ke+
     *            keu,nsi)
             u02 = xss(liu+((nx4-1)*nx32)+nx4*(ji-1)*nx32+nx32*ke+
     *            keu,nsi)
          endif
          if (e.eq.ei1) then
             enew = e01
             ulab = u01
             go to 4500
          endif
          if (e.eq.ei2) then
             enew = e02
             ulab = u02
             go to 4500
          endif
          if (e.ne.ei1.or.e.ne.ei2) then
             rho  = (e-ei1)/(ei2-ei1)
             enew = e01 + rho*(e02-e01)
             ulab = u01 + rho*(u02-u01)
          endif
       else
          if (ji.eq.0) then
            enew = xss(liu+kk*nx32,nsi)
            ulab = xss(liu+kk*nx32+keu,nsi)
          endif
          if (ji.eq.nein) then
            enew = xss(liu+((nx4-1)*nx32)+nx4*(nein-2)*nx32+nx32*ke,
     *            nsi)
            ulab = xss(liu+((nx4-1)*nx32)+nx4*(nein-2)*nx32+nx32*ke+
     *            keu,nsi)
          endif
       endif
c*****************calculate new direction cosines**********************
4500  if (lsmpl.eq.1) go to 4520
      fmu  = ulab
      sinpsi = sqrt(1.0-fmu*fmu)
      call  azirn(sineta,coseta)
      stheta = sqrt(v**2+w**2)
      if (stheta.le.0.0) then
        cosphi = 1.0
        sinphi = 0.0
```

```
      else
        cosphi = v/stheta
        sinphi = w/stheta
      end if
      sinpe  = sinpsi*sineta
      cossin = coseta*sinpsi
      usico  = u*cossin
      v      = v*fmu+cosphi*usico-sinphi*sinpe
      w      = w*fmu+sinphi*usico+cosphi*sinpe
      u      = u*fmu-cossin*stheta
      go to 4550
c*************adjust energy for velocity of neutron*********************
4520  vnr(1) = vnr(1)/snr
      vnr(2) = vnr(2)/snr
      vnr(3) = vnr(3)/snr
      call rotasz(ulab,vnr,u,irt)
      ar     = a/(8.625e-11*temp(ith))
      enorm  = sqrt(enew*ar)
c      vnew2  = 9.5528e17*enew
c      enorm  = sqrt(vnew2)
      vx     = enorm*u + vnuc(1)
      vy     = enorm*v + vnuc(2)
      vz     = enorm*w + vnuc(3)
      vsq    = vx**2 + vy**2 + vz**2
      u      = vx/sqrt(vsq)
      v      = vy/sqrt(vsq)
      w      = vz/sqrt(vsq)
      enew   = vsq/ar
c      enew   = 1.0468e-18 * vsq
      e      = eold
      lsmpl = 0
c************************determine the new energy group******************
4550  if (enew.gt.20.) enew=20.
      if (enew.lt.1.e-11) enew=1.e-11
      if (enew.ge.egp(1))ignew=1
      if (enew.le.egp(ngp1))ignew=ngp
      do 4560 j=1,ngp
        if (enew.lt.egp(j).and.enew.ge.egp(j+1)) then
          ignew=j
          go to 4600
        endif
4560  continue
c*******************************end collision*******************************
4600  tlf    = tlf+ab*tme
      if (lfstart.eq.1) then
        lfstart = 0
        call clear(ithn,23)
        call clear(spl(1,neut),23)
      endif
c
4625  return
```

```
c
4800  call stop('pkeno message number med-3: invalid angular
     1          distribution specified in problem')
10000 format('0keno message number k5-128',10x,
     * ' neutron bank is full.  splitting not allowed')
      end
```

Appendix C

PKENO V.a Input Listing for Benchmark Problems

PKENO V.a Input Listing

for 25 Problem KENO Test Set

```
. . . . . .
sample problem 1  2x2x2 2c8 units
read param flx=yes fdn=yes far=yes end param
read mixt
mix=1 92234 50c 4.82716e-4 293
     92235 50c 4.47971e-2 293
     92236 50c 9.57231e-5 293
     92238 50c 2.65767e-3 293
end mixt
read geometry
cylinder     1 1 5.748 5.3825 -5.3825
cuboid       0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geometry
read array  nux=2 nuy=2 nuz=2 end array
end data
. . . . . .
sample problem 2  2c8 bare with 8 unit types matrix calculation
read param flx=yes fdn=yes
mku=yes fmu=yes mkp=yes fmp=yes
end param  read geometry  unit 1
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 2
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 3  cylinder  1 1 5.748 5.3825 -5.3825 cuboid  0 1 6.87 -6.87 6.87
-6.87 6.505 -6.505
box type 4  cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
box type  5
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 6
cylinder  1 1 5.748 5.3825 -5.3825  cuboid  0 1 6.87 -6.87 6.87 -6.87
6.505 -6.505 unit 7  cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 8
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505 end geom
read mixt
mix=1 92234 50c 4.82716e-04 293 92235 50c 4.47971e-02 293 92236 50c 9.57231e-05
293 92238 50c 2.65767e-03 293
end mixt
read array  nux=2 nuy=2 nuz=2  loop  10*1 3*2 7*1  3 1 1 1 2 2 1
1 1 1  4 2 2 1 2 2 1 1 1 1  5 6*1 2 2 1  6 2 2 1 1 1 1 2 2 1
7  1 1 1 2 2 1 2 2 1  8 2 2 1 2 2 1 2 2 1 end array
end data
```

```
. . . . . .
sample problem 3  2c8  15.24 cm paraffin refl
read param flx=yes fdn=yes pwt=yes end param
read array nux=2 nuy=2 nuz=2 end array
read mixt
sab=2 poly 01t
mix=1 92234 50c 4.82716e-04 293.0 92235 50c 4.47971e-02 293.0
92236 50c 9.57231e-05 293.0 92238 50c 2.65767e-03 293.0
mix=2 6012 50c 3.97311e-02 293.0 1001 50c 8.26407e-02 293.0
end mixt
read geom
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 11.74 -11.74 11.74 -11.74 11.375 -11.375
core  0 1 -23.48 -23.48 -22.75
cuboid  2 2 26.48 -26.48 26.48 -26.48 25.75 -25.75
cuboid  2 3 29.48 -29.48 29.48 -29.48 28.75 -28.75
cuboid  2 4 32.48 -32.48 32.48 -32.48 31.75 -31.75
cuboid  2 5 35.48 -35.48 35.48 -35.48 34.75 -34.75
cuboid  2 6 38.72 -38.72 38.72 -38.72 37.99 -37.99
end geom
read bias  id=400 2 6  end bias  end data
. . . . . .
sample problem 4  2c8 15.24 cm paraffin refl automatic refl
read param pwt=yes flx=yes fdn=yes end param
read geometry  cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 11.74 -11.74 11.74 -11.74 11.375 -11.375
core  0 1 -23.48 -23.48 -22.75
reflector  2 2 6*3.0 5
reflector  2 7 6*.24 1
end geom
read mixt
sab=2 poly 01t
mix=1 92234 50c 4.82716e-04 293
      92235 50c 4.47971e-02 293
      92236 50c 9.57231e-05 293
      92238 50c 2.65767e-03 293
mix=2  6012 50c 3.97311e-02 293
       1001 50c 8.26407e-02 293
end mixt
read arra  nux=2 nuy=2 nuz=2 end array
read bias  id=400 2 7  end bias
end data
end
. . . . . .
```

```
sample problem 5  2c8 12 inch paraffin reflector
read para flx=yes far=yes fdn=yes end para
read array  nux=2 nuy=2 nuz=2 end array
read mixt
sab=2 poly 01t
mix=1 92234 50c 4.82716e-04 293
      92235 50c 4.47971e-02 293
      92236 50c 9.57231e-05 293
      92238 50c 2.65767e-03 293
mix=2  6012 50c 3.97311e-02 293
       1001 50c 8.26407e-02 293
end mixt
read geom
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 11.74 -11.74 11.74 -11.74 11.375 -11.375
core  0 1 -23.48 -23.48 -22.75
cuboid  2 2 26.48 -26.48 26.48 -26.48 25.75 -25.75
cuboid  2 3 29.48 -29.48 29.48 -29.48 28.75 -28.75
cuboid  2 4 32.48 -32.48 32.48 -32.48 31.75 -31.75
cuboid  2 5 35.48 -35.48 35.48 -35.48 34.75 -34.75
cuboid  2 6 38.48 -38.48 38.48 -38.48 37.75 -37.75
cuboid  2 7 41.48 -41.48 41.48 -41.48 40.75 -40.75
cuboid  2 8 44.48 -44.48 44.48 -44.48 43.75 -43.75
cuboid  2 9 47.48 -47.48 47.48 -47.48 46.75 -46.75
cuboid  2 10 50.48 -50.48 50.48 -50.48 49.75 -49.75
cuboid  2 11 53.96 -53.96 53.96 -53.96 53.23 -53.23
end geom
read bias  id=400 2 11  end bias
end data
end
. . . . . .
sample problem 6  one 2c8 unit (single unit)
read param flx=yes fdn=yes far=yes end param
read mixt
mix=1 92234 50c 4.82716e-4 293
      92235 50c 4.47971e-2 293
      92236 50c 9.57231e-5 293
      92238 50c 2.65767e-3 293
end mixt
read geometry
cylinder     1 1 5.748 5.3825 -5.3825
end geometry
end data
. . . . . .
```

```
sample problem 7  bare 2c8 using specular reflection
read para flx=yes fdn=yes far=yes end parameters
read mixt
mix=1 92234 50c 4.82716e-4 293
     92235 50c 4.47971e-2 293
     92236 50c 9.57231e-5 293
     92238 50c 2.65767e-3 293
end mixt
read geom  cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geom
read bounds +fc=specular end bounds  end data
end
. . . . . .
sample problem 8  infinitely long cylinder from 2c8 unit
read mixt
mix=1 92234 50c 4.82716e-4 293
     92235 50c 4.47971e-2 293
     92236 50c 9.57231e-5 293
     92238 50c 2.65767e-3 293
end mixtures
read geometry  cylinder  1 1 5.748 10.0 -10.0
cuboid  0 1 6.87 -6.87 6.87 -6.87 10.0 -10.0
end geometry
read bounds zfc=mirror  end bounds
end data
end
. . . . . .
sample problem 9  infinite array of 2c8 units
read mixtures
mix=1 92234 50c 4.82716e-4 293
     92235 50c 4.47971e-2 293
     92236 50c 9.57231e-5 293
     92238 50c 2.65767e-3 293
end mixt
read boun  all=mir  end boun
read geom
cylinder  1 1 5.748 5.3825 -5.3825  cuboid  0 1 6.87 -6.87 6.87 -6.87
6.505 -6.505  end geom  end data
end
. . . . . .
```

```
sample problem 10  case 2c8 bare  write restart
read parameters flx=yes fdn=yes far=yes res=5 wrs=94
end parameters
read mixt
mix=1 92234 50c 4.82716e-04 293
      92235 50c 4.47971e-02 293
      92236 50c 9.57231e-05 293
      92238 50c 2.65767e-03 293
end mixtures
read geometry
cylinder 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geometry
read array  nux=2 nuy=2 nuz=2  end array
end data
end
. . . . . .
sample problem 11  2c8 bare  read restart data
read param  beg=51  rst=94 res=0 end param
end data
end
. . . . . .
sample problem 12 4 aqueous 4 metal mixed units
read param lng=2500000 flx=yes fdn=yes nub=yes
smu=yes mkp=yes mku=yes fmp=yes fmu=yes end param
read mixt
sab=2 lwtr 01t
mix=1 92234 50c 4.82716e-04 293
      92235 50c 4.47971e-02 293
      92236 50c 9.57231e-05 293
      92238 50c 2.65767e-03 293
mix=2 1001  50c 5.77964e-02 293
      7014  50c 2.13092e-03 293
      8016  50c 3.74130e-02 293
      92234 50c 1.06784e-05 293
      92235 50c 9.84599e-04 293
      92236 50c 5.29385e-06 293
      92238 50c 6.19413e-05 293
mix=3 1001  50c 5.68187e-02 293
      6012  50c 3.55117e-02 293
      8016  50c 1.42047e-02 293
end mixt
read geom
box type 1
cylinder  2 1 9.525 8.89 -8.89
cylinder  3 1 10.16 9.525 -9.525
cuboid  0 1 10.875 -10.875 10.875 -10.875 10.24 -10.24
box type  2
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 6.59 -15.16 6.225 -14.255
box type  3
```

```
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 15.16 -6.59 6.225 -14.255
box type  4
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 6.59 -15.16 14.255 -6.225
box type  5
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 15.16 -6.59 14.255 -6.225
end geom  read array  nux=2 nuy=2 nuz=2  loop
1 3r2 1 2 1 1 2 1  2 9r1  3 3r1 2 2 1 3r1  4 6r1 2 2 1 5 3r1 2 2 1 2 2 1
end array
end data
end
. . . . . .
sample problem 13  two cuboids in a cylindrical annulus
read geom
unit 1
cuboid  1 1 6.35 -6.35 6.35 -6.35 7.62 0.0
cylinder  0 1 13.97 7.62 0.0 orig -6.0934 0.0
cylinder  1 1 19.05 7.62 0.0 orig -6.0934 0.0
cuboid  0 1 12.9566 -25.1434 19.05 -19.05 7.62 0.0
unit 2
cuboid  1 1 6.35 -6.35 6.35 -6.35 8.56 0.0
cylinder  0 1 13.97 8.56 0.0 origin 6.0934 0.0
cylinder  1 1 19.05 8.56 0.0 origin 6.0934 0.0
cuboid  0 1 25.1434 -12.9566 19.05 -19.05 8.56 0.0
unit 3
cuboid  1 1 6.35 -6.35 6.35 -6.35 2.616 0.0
cuboid  0 1 25.1434 -12.9566 19.05 -19.05 2.616 0.0
end geom
read mixt
mix=1  92234 50c 4.80915e-04 293
       92235 50c 4.46299e-02 293
       92236 50c 9.53659e-05 293
       92238 50c 2.64775e-03 293
end mixt
read array nux=1 nuy=1 nuz=3  fill 1 2 3 t end array end data
. . . . . .
sample problem 14  u metal cylinder in an annulus
read mixt
mix=1  92234 50c 4.80915e-04 293
       92235 50c 4.46299e-02 293
       92236 50c 9.53659e-05 293
       92238 50c 2.64775e-03 293
end mixt
read geom
cylinder  1 1 8.89 10.109 0.0 orig 5.0799 0.0
cylinder  0 1 13.97 10.109 0.0
cylinder  1 1 19.05 10.109 0.0
end geom
end data
```

```
. . . . . .
sample problem 15  small water reflected sphere on plexiglas collar
read param tme=90 tba=2 lng=1500000 flx=yes fdn=yes end param
read mixt
sab=3 lwtr 01t
mix=1  92234 50c 5.65801e-04 293
       92235 50c 4.70211e-02 293
       92236 50c 9.58966e-05 293
       92238 50c 4.65935e-04 293
mix=2   1001 50c 5.68187e-02 293
        6012 50c 3.55117e-02 293
        8016 50c 1.42047e-02 293
mix=3   1001 50c 6.67514e-02 293
mix=3   8016 50c 3.33757e-02 293
end mixt
read geom
unit 1
hemisphe-z 1 1 6.5537 chord -5.09066
cylinder  3 1 4.1275 -5.09066 -7.63065
cylinder  2 1 12.7   -5.09066 -7.63065
cuboid  3 1 4p12.7 -5.09066 -7.63065
unit 2
hemisphe+z 1 1 6.5537 chord 5.09066
cuboid  3 1 4p12.7 6.5537 -5.09066
core  0 1 -12.7 -12.7 -7.092175
cylinder  3 1 17.97 2p7.0922
replicate 3 2 3*3.0 5
end geom
read bias  id=500 2 6 end bias
read array nux=1 nuy=1 nuz=2  fill 1 2 end array
read plot ttl='x-z slice through the center of the sphere' lpi=6
xul=-20.0 zul=10.0 yul=0.0  xlr=20.0 ylr=0.0 zlr=-10.0
uax=1.0 wdn=-1.0 nax=130    nch=' *0-'
end plot
end data
end
. . . . . .
sample problem 16 uo2f2 infinite slab k-infinity
read parameters lng=2500000 end parameters
read mixt
sab=1 lwtr 01t
sab=3 lwtr 01t
mix=1   9019 50c 2.96286e-03 293
        1001 50c 6.04824e-02 293
        8016 50c 3.32041e-02 293
       92235 50c 1.38188e-03 293
       92238 50c 9.95503e-05 293
mix=2  11023 50c 2.39502e-03 293
       13027 50c 4.97719e-04 293
       14000 50c 1.80267e-02 293
        5010 50c 9.08241e-04 293
```

```
       5011 56c 3.68719e-03 293
       8016 50c 4.49173e-02 293
 mix=3   9019 50c 2.96286e-03 293
       1001 50c 6.04824e-02 293
       8016 50c 3.32041e-02 293
      92235 50c 1.38188e-03 293
      92238 50c 9.95503e-05 293
       5010 50c 2.92803e-04 293
       5011 56c 1.18870e-03 293
end mixt
read geometry
cuboid 1 1 2.479 -2.479 100 -100 100 -100
cuboid 2 1 3.749 -3.749 100 -100 100 -100
cuboid 3 1 17.479 -17.479 100 -100 100 -100
end geom
read bounds  all=mirror  end bounds    read array  end array
end data
end
. . . . . .
sample problem 17 93% uo2f2 solution sphere forward calculation
read mixt
sab=1  lwtr 01t
mix=1   1001 50c 6.54785e-02 293
       8016 50c 3.34202e-02 293
       9019 50c 6.80923e-04 293
      92235 50c 3.16909e-04 293
      92238 50c 2.35521e-05 293
end mixt
read geometry
sphere 1 1 16.0
end geom
end data
end
. . . . . .
sample problem 18   1f27 demonstration of options problem
read para  lng=3000000 fdn=yes nub=yes
  mku=yes fmu=yes mkh=yes fmh=yes mka=yes fma=yes rnd=f12c09ed2195
  pwt=yes far=yes flx=yes pgm=yes
 end para
read mixt
  sab=1 lwtr 01t
  sab=3 poly 01t
  sab=4 lwtr 01t
  sab=5 lwtr 01t
  mix= 1  1001 50c 5.77964e-02 293
        7014 50c 2.13092e-03 293
        8016 50c 3.74130e-02 293
       92234 50c 1.06784e-05 293
       92235 50c 9.84599e-04 293
       92236 50c 5.29385e-06 293
       92238 50c 6.19413e-05 293
```

```
  mix=2   1001 50c 5.68187e-02 293
          6012 50c 3.55117e-02 293
          8016 50c 1.42047e-02 293
  mix=3   6012 50c 3.97311e-02 293
          1001 50c 8.26407e-02 293
  mix=4   8016 50c 3.33757e-11 293
          1001 50c 6.67514e-11 293
  mix=5   8016 50c 3.3371e-2   293
          1001 50c 6.6674e-2   293
 end mixt
read geom unit 1 cylinder 1 1 9.52 8.7804 -8.7804
 cylinder 0 1 9.52 8.9896 -8.7804
 cylinder 2 1 10.16 9.6296 -9.4204
 cuboid 4 1 18.45 -18.45 18.45 -18.45 17.8946 -17.6854
 unit 2 array 1 3*0.0
 unit 3 array 2 3*0.0
 unit 4 array 3 3*0.0
 unit 5 array 4 3*0.0
 global
 unit 6 cuboid 4 1 55.3501 -55.3501 55.3501 -55.3501 53.3701 -53.3701
  hole 2 -55.35   -18.45   -17.79
  hole 3 -55.35   -18.45   -53.3701
  hole 4  18.4501 -18.45   -53.3701
  hole 5 -55.3501 -55.3501 -53.3701
 replicate 3 2 6*3 5  replicate 3 7 6*0.24 1
 cuboid 5 1 70.5901 -70.5901 70.5901 -70.5901 68.6101 -99.0901 end geom
 read bias  id=400 2 7  end bias
read array
 ara=1 nux=2 nuy=2 nuz=2 fill f1 end fill
 ara=2 nux=2 nuy=2 nuz=1 fill f1 end fill
 ara=3 nux=1 nuy=2 nuz=3 fill f1 end fill
 ara=4 nux=3 nuy=1 nuz=3 fill f1 end fill
 end array
 read start nst=6 tfx=0.0 tfy=0.0 tfz=0.0
 lnu=300 ps6=yes
 end start
read plot          plt=yes lpi=6
ttl=?  1f27 xy plot at z=0.0 ?  xul=-71.0 yul=71.0 zul=0.0
 xlr=71.0 ylr=-71.0 zlr=0.0 uax=1 vdn=-1 nax=130 nch=?.*-3 ?
 run=yes end       ttl=?unit map 1f27 xy plot at z=0.0?
 pic=unit nch=? 123456? end plot     end data
end
. . . . . .
sample problem 19 4 aqueous 4 metal array of arrays (samp prob 12)
read param lng=2500000 flx=yes fdn=yes nub=yes smu=yes mkp=yes
mku=yes fmp=yes fmu=yes end param
read mixt
sab=2 lwtr 01t
mix=1 92234 50c 4.82716e-04 293
      92235 50c 4.47971e-02 293
      92236 50c 9.57231e-05 293
```

```
      92238 50c 2.65767e-03 293
 mix=2 1001  50c 5.77964e-02 293
      7014  50c 2.13092e-03 293
      8016  50c 3.74130e-02 293
      92234 50c 1.06784e-05 293
      92235 50c 9.84599e-04 293
      92236 50c 5.29385e-06 293
      92238 50c 6.19413e-05 293
 mix=3 1001  50c 5.68187e-02 293
      6012  50c 3.55117e-02 293
      8016  50c 1.42047e-02 293
 end mixt
 read geom
 unit 1
 com='uranyl nitrate solution in a plexiglas container'
 cylinder  2 1 9.525 2p8.89
 cylinder  3 1 10.16 2p9.525
 cuboid  0 1 4p10.875 2p10.24
 unit 2
 com='uranium metal cylinder'
 cylinder  1 1 5.748 2p5.3825
 cuboid  0 1 4p6.59 2p6.225
 unit 3
 com='1x2x2 array of solution units'
 array 1 3*0.0
 unit 4
 com='1x2x2 array of metal units padded to match solution array'
 array 2 3*0.0
 replicate 0 1 2*0.0 2*8.57 2*8.03 1
 end geom
 read array ara=1 nux=1 nuy=2 nuz=2 fill f1 end fill
 ara=2 nux=1 nuy=2 nuz=2 fill f2 end fill gbl=3 ara=3 nux=2 nuy=1 nuz=1
 com='composite array of solution and metal units'
 fill 4 3 end fill
 end array
 end data
 end
 . . . . . .
 sample problem 20 triangular pitched array
 read mixt
 sab=1  lwtr 01t
 mix=1  92235 50c 1.37751e-03 293
       92238 50c 9.92354e-05 293
        8016 50c 3.32049e-02 293
        9019 50c 2.95349e-03 293
        1001 50c 6.05028e-02 293
 mix=2  13027 50c 6.02374e-02 293
 end mixt
 read geom
 unit 1
 cylinder   1 1 10.16 18.288 0
```

```
cylinder   2 1 10.312 18.288 -.152
unit 2
cuboid     0 1 4p50 50 -.152
hole       1 3r0
hole       1 21.006 2r0
hole       1 -21.006 2r0
hole       1 10.503 18.192 0
hole       1 -10.503 18.192 0
hole       1 10.503 -18.192 0
hole       1 -10.503 -18.192 0
end geom
read array nux=1 nuy=1 nuz=1 fill 2 end fill end array
read plot ttl='hex array' pic=mix lpi=6
xul=0 yul=100 zul=10
xlr=100 ylr=0 zlr=10 uax=1 vdn=-1 nax=120
nch=' 12' end plot
end data
end
. . . . . .
sample problem 21  partially filled sphere
read geom
hemisphe-z  1 1 34.6   chord 30.
sphere      0 1 34.6
sphere      2 1 34.759
end geom
read mixt
sab=1  lwtr  01t
mix=1  1001  50c 6.15670e-02 293
       8016  50c 3.32845e-02 293
       9019  50c 2.50098e-03 293
       92234 50c 2.54223e-07 293
       92235 50c 6.18922e-05 293
       92238 50c 1.18834e-03 293
mix=2  13027 50c 6.02374e-02 293
end mixt
end data
. . . . . .
sample problem 22   case 2c8 bare with 3 nested, equal volume holes
read parameters flx=yes fdn=yes far=yes end parameters
read mixt
mix=1 92234 50c 4.82716e-4 293
      92235 50c 4.47971e-2 293
      92236 50c 9.57231e-5 293
      92238 50c 2.65767e-3 293
end mixt
read geometry
unit 1
cylinder 1 1 3.621 2p3.3907
unit 2
cylinder 1 1 4.5622 2p4.2721
hole 1 3*0.0
```

```
unit 3
cylinder 1 1 5.2224 2p4.8903
hole 2 3*0.0
unit 4
cylinder 1 1 5.748 5.3825 -5.3825
hole 3 3*0.0
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geometry
read array  nux=2 nuy=2 nuz=2 fill f4 end fill end array
end data
. . . . . .
sample problem 23  case 2c8 bare as mixed zhemicylinders
read parameters fdn=yes end parameters
read mixt
mix=1 92234 50c 4.82716e-4 293
     92235 50c 4.47971e-2 293
     92236 50c 9.57231e-5 293
     92238 50c 2.65767e-3 293
end mixt
read geometry
unit 1
com='-x half of unit 3'
zhemicyl-x 1 1 5.748 5.3825 -5.3825
cuboid  0 1 0.0  -6.87 6.87 -6.87 6.505 -6.505
unit 2
com='+x half of unit 3'
zhemicyl+x 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87  0.0  6.87 -6.87 6.505 -6.505
unit 3
com='cylinder composed of equal halves (zhemicylinders with x radii)'
array 1 3*0.0
unit 4
com='-x portion (more than half) of unit 6'
zhemicyl-x 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 3.0  -6.87 6.87 -6.87 6.505 -6.505
unit 5
com='+x portion (less than half) of unit 6'
zhemicyl+x 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.87  3.0  6.87 -6.87 6.505 -6.505
unit 6
com='cylinder composed of unequal halves (zhemicylinders with x radii)'
array 2 3*0.0
unit 7
com='cylinder of a single zhemicylinder in the -x direction'
zhemicyl-x 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 8
com='cylinder of a single zhemicylinder in the +x direction'
zhemicyl+x 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 9
```

```
com='-y half of unit 11'
zhemicyl-y 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 0.0  -6.87 6.505 -6.505
unit 10
com='+y half of unit 11'
zhemicyl+y 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87  0.0  6.505 -6.505
unit 11
com='cylinder composed of equal halves (zhemicylinders with z radii)'
array 3 3*0.0
unit 12
com='-y portion (more than half) of unit 14'
zhemicyl-y 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 6.87 -6.87 3.0  -6.87 6.505 -6.505
unit 13
com='+y portion (less than half) of unit 14'
zhemicyl+y 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.87 -6.87 6.87  3.0  6.505 -6.505
unit 14
com='cylinder composed of unequal halves (zhemicylinders with z radii)'
array 4 3*0.0
unit 15
com='cylinder of a single zhemicylinder in the -y direction'
zhemicyl-y 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 16
com='cylinder of a single zhemicylinder in the +y'
zhemicyl+y 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geometry
read array
com='array 1 defines unit 3 (zhemicylinders with x radii)'
ara=1 nux=2 nuy=1 nuz=1 fill 1 2 end fill
com='array 2 defines unit 6 (zhemicylinders with x radii)'
ara=2 nux=2 nuy=1 nuz=1 fill 4 5 end fill
com='array 3 defines unit 11 (zhemicylinders with y radii)'
ara=3 nux=1 nuy=2 nuz=1 fill 9 10 end fill
com='array 4 defines unit 14 (zhemicylinders with y radii)'
ara=4 nux=1 nuy=2 nuz=1 fill 12 13 end fill
com='array 5 defines the total 2c8 problem'
gbl=5 ara=5 nux=2 nuy=2 nuz=2 fill 3 7 6 8 11 15 14 16 end fill
end array
end data
. . . . . .
sample problem 24  case 2c8 bare as mixed xhemicylinders
read parameters fdn=yes end parameters
read mixt
mix=1 92234 50c 4.82716e-4 293
      92235 50c 4.47971e-2 293
      92236 50c 9.57231e-5 293
      92238 50c 2.65767e-3 293
```

```
end mixt
read geometry
unit 1
com='-y half of unit 3'
xhemicyl-y 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.505 -6.505 0.0 -6.87 6.87 -6.87
unit 2
com='+y half of unit 3'
xhemicyl+y 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.505 -6.505 6.87 0.0  6.87 -6.87
unit 3
com='cylinder composed of equal halves (xhemicylinders with y radii)'
array 1 3*0.0
unit 4
com='-y portion (more than half) of unit 6'
xhemicyl-y 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 6.505 -6.505 3.0 -6.87 6.87 -6.87
unit 5
com='+y portion (less than half) of unit 6'
xhemicyl+y 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.505 -6.505 6.87 3.0  6.87 -6.87
unit 6
com='cylinder composed of unequal halves (xhemicylinders with y radii)'
array 2 3*0.0
unit 7
com='cylinder of a single xhemicylinder in the -y direction'
xhemicyl-y 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 -6.87
unit 8
com='cylinder of a single xhemicylinder in the +y direction'
xhemicyl+y 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 -6.87
unit 9
com='-z half of unit 11'
xhemicyl-z 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.505 -6.505 6.87 -6.87 0.0 -6.87
unit 10
com='+z half of unit 11'
xhemicyl+z 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 0.0
unit 11
com='cylinder composed of equal halves (xhemicylinders with z radii)'
array 3 3*0.0
unit 12
com='-z portion (more than half) of unit 14'
xhemicyl-z 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 6.505 -6.505 6.87 -6.87 3.0 -6.87
unit 13
com='+z portion (less than half) of unit 14'
xhemicyl+z 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 3.0
```

```
unit 14
com='cylinder composed of unequal halves (xhemicylinders with z radii)'
array 4 3*0.0
unit 15
com='cylinder of a single xhemicylinder in the -z direction'
xhemicyl-z 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 -6.87
unit 16
com='cylinder of a single xhemicylinder in the +z direction'
xhemicyl+z 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 -6.87
end geometry
read array
com='array 1 defines unit 3 (xhemicylinders with y radii)'
ara=1 nux=1 nuy=2 nuz=1 fill 1 2 end fill
com='array 2 defines unit 6 (xhemicylinders with y radii)'
ara=2 nux=1 nuy=2 nuz=1 fill 4 5 end fill
com='array 3 defines unit 11 (xhemicylinders with z radii)'
ara=3 nux=1 nuy=1 nuz=2 fill 9 10 end fill
com='array 4 defines unit 14 (xhemicylinders with z radii)'
ara=4 nux=1 nuy=1 nuz=2 fill 12 13 end fill
com='array 5 defines the total 2c8 problem'
gbl=5 ara=5 nux=2 nuy=2 nuz=2 fill 3 7 6 8 11 15 14 16 end fill
end array
end data
. . . . . .
sample problem 25  case 2c8 bare as mixed yhemicylinders
read parameters fdn=yes end parameters
read mixt
mix=1 92234 50c 4.82716e-4 293
      92235 50c 4.47971e-2 293
      92236 50c 9.57231e-5 293
      92238 50c 2.65767e-3 293
end mixt
read geometry
unit 1
com='-x half of unit 3'
yhemicyl-x 1 1 5.748 5.3825 -5.3825
cuboid  0 1 0.0  -6.87 6.505 -6.505 6.87 -6.87
unit 2
com='+x half of unit 3'
yhemicyl+x 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87  0.0  6.505 -6.505 6.87 -6.87
unit 3
com='cylinder composed of equal halves (yhemicylinders with x radii)'
array 1 3*0.0
unit 4
com='-x portion (more than half) of unit 6'
yhemicyl-x 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 3.0  -6.87 6.505 -6.505 6.87 -6.87
unit 5
```

```
com='+x portion (less than half) of unit 6'
yhemicyl+x 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.87  3.0  6.505 -6.505 6.87 -6.87
unit 6
com='cylinder composed of unequal halves (yhemicylinders with x radii)'
array 2 3*0.0
unit 7
com='cylinder of a single yhemicylinder in the -x direction'
yhemicyl-x 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 -6.87
unit 8
com='cylinder of a single yhemicylinder in the +x direction'
yhemicyl+x 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 -6.87
unit 9
com='-z half of unit 11'
yhemicyl-z 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.505 -6.505 0.0 -6.87
unit 10
com='+z half of unit 11'
yhemicyl+z 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 0.0
unit 11
com='cylinder composed of equal halves (yhemicylinders with z radii)'
array 3 3*0.0
unit 12
com='-z portion (more than half) of unit 14'
yhemicyl-z 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 6.87 -6.87 6.505 -6.505 3.0 -6.87
unit 13
com='+z portion (less than half) of unit 14'
yhemicyl+z 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 3.0
unit 14
com='cylinder composed of unequal halves (yhemicylinders with z radii)'
array 4 3*0.0
unit 15
com='cylinder of a single  yhemicylinder in the -z direction'
yhemicyl-z 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 -6.87
unit 16
com='cylinder of a single yhemicylinder in the +z direction'
yhemicyl+z 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 -6.87
end geometry
read array
com='array 1 defines unit 3 (yhemicylinders with x radii)'
ara=1 nux=2 nuy=1 nuz=1 fill 1 2 end fill
com='array 2 defines unit 6 (yhemicylinders with x radii)'
ara=2 nux=2 nuy=1 nuz=1 fill 4 5 end fill
com='array 3 defines unit 11 (yhemicylinders with z radii)'
```

```
ara=3 nux=1 nuy=1 nuz=2 fill 9 10 end fill
com='array 4 defines unit 14 (zhemicylinders with z radii)'
ara=4 nux=1 nuy=1 nuz=2 fill 12 13 end fill
com='array 5 defines the total 2c8 problem'
gbl=5 ara=5 nux=2 nuy=2 nuz=2 fill 3 7 6 8 11 15 14 16 end fill
end array
end data
```

PKENO V.a Input Listing for

MCNP Benchmark Problems

```
. . . . . .
(mcnp primer ex.1) plutonium sphere coated with ni
read para npg=1500 end para
read mixt
mix=1 94239 50c 3.7047e-2 293
     94240 50c 1.7510e-3 293
     94241 50c 1.1700e-4 293
mix=2 28000 50c 9.1322e-2 293
end mixt
read geometry
sphere    1 1 6.38493
sphere    2 1 6.39763
end geometry
end data
. . . . . .
(mcnp primer ex.2.1) bare plutonium cylinder
read param npg=1500 end param
read mixt
mix=1 94239 50c 3.9802e-2 293
end mixt
read geometry
cylinder    1 1 4.935 17.273 0.0
end geometry
end data
. . . . . .
(mcnp primer ex.2.2) plutonium cylinder reflected with nat. u
read param npg=1500 end param
read mixt
mix=1 94239 50c 3.9802e-2 293
mix=2 92238 50c 4.7219e-2 293
     92235 50c 3.4246e-4 293
end mixt
read geometry
cylinder    1 1 4.935 6.909 0.0
cylinder    2 1 9.935 6.909 0.0
end geometry
end data
. . . . . .
(mcnp primer ex.2.3) plutonium cylinder reflected with nat. u
read param npg=1500 end param
read mixt
mix=1 94239 50c 3.9802e-2 293
mix=2 92238 50c 4.7219e-2 293
     92235 50c 3.4246e-4 293
end mixt
read geometry
cylinder    1 1 4.935 6.909 0.0
```

```
cylinder    2 1 9.935 11.909 -5.0
end geometry
end data
. . . . . .
(mcnp primer ex3) s(a,b) treatment mcnp primer
read mixt
sab=1  lwtr  01t
mix=1  1001  50c 5.7058e-02 293
       8016  50c 3.2929e-02 293
       9019  50c 4.3996e-03 293
       92235 50c 1.0889e-04 293
       92238 50c 2.0909e-03 293
mix=2  13027 50c 6.02374e-02 293
end mixt
read geom
cylinder    1 1 20.12   100.0 0.0
cylinder    0 1 20.12   110.0 0.0
cylinder    2 1 20.2787 110.0 -0.1587
end geom
end data
. . . . . .
(mcnp primer ex. 4) 2 uo2f2 cylinders in tank of water
read mixt
sab=1 lwtr 01t
sab=3 lwtr 01t
mix=1 92235 50c 1.1760e-3 293
      92238 50c 8.2051e-5 293
       8016 50c 3.3621e-2 293
       1001 50c 6.221e-2  293
       9019 50c 2.5161e-3 293
mix=2 13027 50c 6.0263e-2 293
mix=3  1001 50c 6.6857e-2 293
       8016 50c 3.3428e-2 293
end mixt
read geometry
unit 1
cylinder   1 1  6.35  70.2  0.0
cylinder   2 1  6.50  80.0 -0.15
cuboid     3 1  8.5  -16.5  26.5 -26.5 80.0 -20.15
unit 2
cylinder   1 1  6.35  70.2  0.0
cylinder   2 1  6.50  80.0 -0.15
cuboid     3 1  26.5 -8.5 26.5 -26.5 80.0 -20.15
global unit 3
array      1   -30.0 -26.0 -50.075
end geometry
read array nux=2 nuy=1 nuz=1
fill 1 2 end fill
end array
end data
end
```

```
. . . . . .
(mcnp primer ex. 5) 3x2 array of pu(no3)4 cylinders
read para npg=1500 lng=2000000 end para
read mixt
sab=1 lwtr 01t
mix=1 94239 50c 2.7682e-4 293
     94240 50c 1.2214e-5 293
     94241 50c 8.3390e-7 293
     94242 50c 4.5800e-8 293
      1001 50c 6.0070e-2 293
      8016 50c 3.6540e-2 293
      7014 50c 2.3699e-3 293
mix=2 26000 50c 6.3310e-2 293
     24000 50c 1.6540e-2 293
     28000 50c 6.5100e-3 293
end mixt
read geometry
unit 1
cylinder   1 1  12.49 39.24  0.0
cylinder   0 1  12.49 101.7  0.0
cylinder   2 1  12.79 102.7 -1.0
cuboid     0 1  17.79 -17.79 17.79 -17.79 102.7 -1.0
global unit 2
array      1   -53.37 -35.58 -51.85
end geometry
read array nux=3 nuy=2 nuz=1
fill 1 1 1 1 1 1 end fill
end array
end data
end
. . . . . .
p21:  jezebel at 95.5% pu-239
read para npg=1500 end para
read mixt
mix=1 94239 50c 3.7555e-2 293
     94240 50c 1.7622e-3 293
end mixt
read geometry
sphere     1 1 6.385
end geometry
end data
end
. . . . . .
p22:  uranium cylinder with 14.11% u-235
read mixt
mix=1 92235 50c 6.6555e-3 293
     92238 50c 4.0002e-2 293
end mixt
read geometry
cylinder     1 1 26.65 44.239 0.
end geometry
```

```
end data
end
. . . . . .
p31:  graphite reflected uranium sphere 93.3% u-235
read para npg=1500 end param
read mixt
sab=2 grph 01t
mix=1 92235 50c 4.5037e-2 293
      92238 50c 3.0914e-3 293
mix=2  6012 50c 8.3389e-2 293
      26000 50c 6.1227e-5 293
      16032 50c 5.0181e-5 293
end mixt
read geometry
sphere     1 1 7.39840
sphere     2 1 12.4984
end geometry
end data
end
. . . . . .
p32:  water reflected uranium sphere 97.67% u-235
read para tme=40 end para
read mixt
sab=2 lwtr 01t
mix=1 92234 50c 5.3000e-4 293
      92235 50c 4.7030e-2 293
      92236 50c 1.0000e-4 293
      92238 50c 4.9000e-4 293
mix=2  1001 50c 6.6790e-2 293
       8016 50c 3.3400e-2 293
end mixt
read geometry
sphere     1 1 6.5537
cylinder   2 1 30.0 35.0 -35.0
end geometry
end data
end
. . . . . .
p41:  3 uo2f2 cylinders in triangle configuration
read para npg=1500 end para
read mixt
sab=1 lwtr 01t
mix=1 92235 50c 2.1936e-4 293
      92238 50c 1.5736e-5 293
       1001 50c 6.7709e-2 293
       8016 50c 3.4291e-2 293
       9019 50c 4.6964e-4 293
mix=2 13027 50c 6.0486e-2 293
end mixt
read geometry
unit 1
```

```
cylinder   1 1  10.15  20.7  -20.7
cylinder   2 1  10.30  20.85 -20.85
global unit 2
cuboid     0 1  20.98 -20.98  30.0 -10.30 20.85 -20.85
hole       1    10.49   0.    0.
hole       1   -10.49   0.    0.
hole       1    0.     18.169 0.
end geometry
end data
end
```

. . . . . .

PKENO V.a Input Listing for

Additional Benchmark Problems

```
. . . . . .
prob.1 water reflected pu metal sphere
read mixt
sab=2 lwtr 01t
mix=1 94239 50c 4.8973e-2 293
mix=2  1001 50c 6.6737e-2 293
       8016 50c 3.3368e-2 293
end mixt
read geometry
sphere     1 1 4.11
sphere     2 1 24.11
end geometry
end data
end
. . . . . .
prob.2 water reflected pu nitrate sphere
read para lng=2000000 end para
read mixt
sab=1  lwtr 01t
sab=3  lwtr 01t
mix=1 94239 50c 3.3662e-4 293
      94240 50c 1.6163e-5 293
       7014 50c 2.7595e-3 293
       1001 50c 6.0260e-2 293
       8016 50c 3.7734e-2 293
mix=2 26000 50c 5.8886e-2 293
      24000 50c 1.7672e-2 293
      28000 50c 8.2374e-3 293
mix=3  1001 50c 6.6063e-2 293
       8016 50c 3.3032e-2 293
end mixt
read geometry
sphere     1 1 14.568
sphere     2 1 14.692
sphere     3 1 39.6922
end geometry
end data
end
. . . . . .
prob.3 bare reflected pu nitrate sphere
read mixt
sab=1 lwtr 01t
mix=1 94239 50c 2.3201e-5 293
      94240 50c 6.0140e-7 293
      94241 50c 1.7720e-8 293
       7014 50c 7.6043e-4 293
       1001 50c 6.4836e-2 293
```

```
      8016 50c 3.4367e-2 293
mix=2 13027 50c 6.0260e-2 293
end mixt
read geometry
sphere    1 1 61.01
sphere    2 1 61.79
end geometry
end data
end
. . . . . .
prob.4 bare u-233 nitrate solution in a sphere
read mixt
sab=1 lwtr 01t
mix=1 92233 50c 4.3293e-5 293
      92234 50c 7.1479e-7 293
      92235 50c 1.7574e-8 293
      92238 50c 2.7763e-7 293
       8016 50c 3.3721e-2 293
       7014 50c 8.8300e-5 293
       1001 50c 6.6747e-2 293
mix=2 13027 50c 6.0263e-2 293
end mixt
read geometry
sphere    1 1 34.595
sphere    2 1 34.9252
end geometry
end data
end
. . . . . .
prob.5 godiva
read param flx=yes fdn=yes far=yes end param
read mixt
mix=1 92234 50c 4.8271e-4 293
      92235 50c 4.4796e-2 293
      92236 50c 9.6028e-5 293
      92238 50c 2.6571e-3 293
end mixt
read geometry
sphere    1 1 8.72
end geometry
end data
. . . . . .
prob.6 bare u-235 nitrate solution in a sphere
read para lng=1500000 end para
read mixt
sab=1 lwtr 01t
mix=1 92235 50c 4.8066e-5 293
      92234 50c 5.3800e-7 293
      92236 50c 1.3800e-7 293
      92238 50c 2.8070e-6 293
       8016 50c 3.3687e-2 293
```

```
      7014 50c 1.8690e-4 293
      1001 50c 6.6228e-2 293
mix=2 13027 50c 6.0263e-2 293
end mixt
read geometry
sphere    1 1 34.595
sphere    2 1 34.9252
end geometry
end data
end
. . . . . .
prob.7 low enriched unmoderated mox sphere
read mixt
sab=2 lwtr 01t
mix=1 94239 50c 1.9591e-3 293
     92235 50c 1.6460e-4 293
     92238 50c 2.2450e-2 293
      8016 50c 4.9148e-2 293
mix=2  1001 50c 6.6680e-2 293
      8016 50c 3.3343e-2 293
end mixt
read geometry
sphere    1 1 19.4
sphere    2 1 39.4
end geometry
end data
end
. . . . . .
prob.8 water reflecte pu-u nitrate in cyl.
read para lng=3000000 end para
read mixt
sab=1 lwtr 01t
sab=3 lwtr 01t
mix=1 94238 50c 8.3767e-9 293
     94239 50c 2.9365e-5 293
     94240 50c 1.7386e-6 293
     94241 50c 1.0752e-7 293
     94242 50c 2.1964e-8 293
     92235 50c 5.0609e-7 293
     92236 50c 1.4077e-8 293
     92238 50c 7.5163e-5 293
      1001 50c 6.3896e-2 293
      8016 50c 3.4957e-2 293
      7014 50c 1.0883e-3 293
mix=2 26000 50c 6.3310e-2 293
     24000 50c 1.6540e-2 293
     28000 50c 6.5100e-3 293
mix=3  1001 50c 6.6680e-2 293
      8016 50c 3.3343e-2 293
end mixt
read geometry
```

```
cylinder   1 1 30.48   95.20 0.0
cylinder   0 1 30.48   106.6  0.0
cylinder   2 1 30.559 106.6 -0.95
cylinder   3 1 45.559 106.6 -20.95
end geometry
end data
end
```

. . . . . .

```
prob.9 water reflecte pu-u-gd nitrate in cyl.
read para lng=3000000 end para
read mixt
sab=1 lwtr 01t
sab=3 lwtr 01t
mix=1 94238 50c 2.3500e-6 293
      94239 50c 1.8260e-4 293
      94240 50c 1.1180e-5 293
      94241 50c 7.1600e-7 293
      94242 50c 1.1900e-7 293
      92234 50c 2.7000e-8 293
      92235 50c 3.0130e-6 293
      92236 50c 5.4000e-8 293
      92238 50c 4.4420e-4 293
       1001 50c 5.5178e-2 293
       8016 50c 3.8688e-2 293
       7014 50c 3.7278e-3 293
      64152 50c 8.1189e-9 293
      64154 50c 8.8496e-8 293
      64155 50c 6.0080e-7 293
      64156 50c 8.3097e-7 293
      64157 50c 6.3530e-7 293
      64158 50c 1.0084e-6 293
      64160 50c 8.8739e-7 293
mix=2 26000 50c 6.3310e-2 293
      24000 50c 1.6540e-2 293
      28000 50c 6.5100e-3 293
mix=3  1001 50c 6.6680e-2 293
       8016 50c 3.3343e-2 293
end mixt
read geometry
cylinder   1 1 30.154  75.3  0.0
cylinder   0 1 30.154 106.6  0.0
cylinder   2 1 30.593 106.6 -0.95
cylinder   3 1 45.593 106.6 -20.95
end geometry
end data
end
```

. . . . . .

```
prob.11 4x4x4 array of pu metal cylinders
read para npg=1500 end para
read mixt
mix=1 94239 50c 4.6053e-2 293
```

```
      94240 50c 2.9263e-3 293
      94241 50c 2.2454e-4 293
      94242 50c 4.8612e-6 293
end mixt
read geometry
unit 1
cylinder   1 1 3.2625 2.3165 -2.3165
cuboid     0 1 6.255 -6.255   6.255 -6.255  3.93 -3.93
global unit 2
array      1  -25.02 -25.02 -15.72
end geometry
read array
ara=1 nux=4 nuy=4 nuz=4
fill 64r1 end fill
end array
end data
end
sheba2
read para lng=2000000 end para
read mixt
sab=1 lwtr 01t
mix=1 92238 50c 2.4990e-3 293
      92235 50c 1.3190e-4 293
      92234 50c 6.8450e-7 293
      92236 50c 1.3160e-6 293
       8016 50c 3.2100e-2 293
       9019 50c 5.3340e-3 293
       1001 50c 5.3540e-2 293
mix=2 26000 50c 5.7107e-2 293
      24000 50c 1.6768e-2 293
      28000 50c 7.4279e-3 293
      25055 50c 1.6706e-3 293
end mixt
read geometry
unit 1
cylinder   0 1  2.36 42.85 0.0
cylinder   2 1  3.0  42.85 0.0
cylinder   1 1  23.9 42.85 0.0
cylinder   2 1  25.4 42.85 0.0
cuboid     0 1  25.4 -25.4 25.4 -25.4 42.85 0.0
unit 2
cylinder   0 1  2.36 73.06 42.85
cylinder   2 1  3.0  73.06 42.85
cylinder   0 1  23.9 73.06 42.85
cylinder   2 1  25.4 73.06 42.85
cuboid     0 1  25.4 -25.4 25.4 -25.4 73.06 42.85
unit 3
cylinder   0 1  2.36 0.0 -1.9
cylinder   2 1  3.0  0.0 -2.54
cylinder   2 1  25.4 0.0 -2.54
cuboid     0 1  25.4 -25.4 25.4 -25.4 0.0 -2.54
```

```
unit 4
cylinder   2 1  25.4 75.6 73.06
cuboid     0 1  25.4 -25.4 25.4 -25.4 75.6 73.06
global unit 5
array      1   -25.4 -25.4 -2.54
end geometry
read array
ara=1  nux=1 nuy=1 nuz=4
fill 3 1 2 4 end fill
end array
end data
end
```

Appendix D

KENO V.a Input Listing for Benchmark Problems

KENO V.a Input Listing

for 25 Problem KENO Test Set

```
. . . . . .
#csas25
sample problem 1  2x2x2 array of 2c8 units
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 1  2x2x2 2c8 units
read parm flx=yes fdn=yes far=yes end parm
read geometry
cylinder     1 1 5.748 5.3825 -5.3825
cuboid       0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geometry
read array nux=2 nuy=2 nuz=2 end array
end data
end
. . . . . .
#csas25
sample problem 2  2c8 bare with 8 unit types matrix calculation
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 1  2x2x2 2c8 units
read param   lng=40000 flx=yes fdn=yes
mku=yes fmu=yes mkp=yes fmp=yes
end param  read geometry  unit 1
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 2
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 3  cylinder  1 1 5.748 5.3825 -5.3825 cuboid  0 1 6.87 -6.87 6.87
-6.87 6.505 -6.505
box type 4  cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
box type  5
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 6
cylinder  1 1 5.748 5.3825 -5.3825  cuboid  0 1 6.87 -6.87 6.87 -6.87
6.505 -6.505 unit 7  cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
```

```
unit 8
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505 end geom
read array  nux=2 nuy=2 nuz=2  loop  10*1 3*2 7*1  3 1 1 1 2 2 1
1 1 1  4 2 2 1 2 2 1 1 1 1  5 6*1 2 2 1  6 2 2 1 1 1 1 2 2 1
7  1 1 1 2 2 1 2 2 1  8 2 2 1 2 2 1 2 2 1 end array
end data
end
```

```
#csas25
sample problem 3  2c8  15.24 cm paraffin refl
27groupndf4                  infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
c      2  0.0 3.97311e-2 293 end
h      2  0.0 8.26407e-2 293 end
end comp
sample problem 3
read param  lng=40000 flx=yes fdn=yes pwt=yes end param
read array nux=2 nuy=2 nuz=2 end array
read geom
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 11.74 -11.74 11.74 -11.74 11.375 -11.375
core  0 1 -23.48 -23.48 -22.75
cuboid  2 2 26.48 -26.48 26.48 -26.48 25.75 -25.75
cuboid  2 3 29.48 -29.48 29.48 -29.48 28.75 -28.75
cuboid  2 4 32.48 -32.48 32.48 -32.48 31.75 -31.75
cuboid  2 5 35.48 -35.48 35.48 -35.48 34.75 -34.75
cuboid  2 6 38.72 -38.72 38.72 -38.72 37.99 -37.99
end geom
read bias  id=400 2 6  end bias
end data
end
```

```
#csas25
sample problem 4  2c8 15.24 cm paraffin refl automatic refl
27groupndf4                  infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
c      2  0.0 3.97311e-2 293 end
h      2  0.0 8.26407e-2 293 end
end comp
sample problem 4  2c8 15.24 cm paraffin refl automatic refl
read param lng=40000 pwt=yes flx=yes fdn=yes end param
read geometry  cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 11.74 -11.74 11.74 -11.74 11.375 -11.375
core  0 1 -23.48 -23.48 -22.75
```

```
reflector  2 2 6*3.0 5
reflector  2 7 6*.24 1
end geom
read arra  nux=2 nuy=2 nuz=2 end array
read bias  id=400 2 7  end bias
end data
end
```

. . . . . .

```
#csas25
sample problem 5  2c8 12 inch paraffin reflector
27groupndf4                  infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
c      2  0.0 3.97311e-2 293 end
h      2  0.0 8.26407e-2 293 end
end comp
sample problem 5  2c8 12 inch paraffin reflector
read para tme=120 tba=2 npg=1500 gen=200 nsk=20
flx=yes far=yes fdn=yes end para
read array  nux=2 nuy=2 nuz=2 end array
read geom
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 11.74 -11.74 11.74 -11.74 11.375 -11.375
core  0 1 -23.48 -23.48 -22.75
cuboid  2 2 26.48 -26.48 26.48 -26.48 25.75 -25.75
cuboid  2 3 29.48 -29.48 29.48 -29.48 28.75 -28.75
cuboid  2 4 32.48 -32.48 32.48 -32.48 31.75 -31.75
cuboid  2 5 35.48 -35.48 35.48 -35.48 34.75 -34.75
cuboid  2 6 38.48 -38.48 38.48 -38.48 37.75 -37.75
cuboid  2 7 41.48 -41.48 41.48 -41.48 40.75 -40.75
cuboid  2 8 44.48 -44.48 44.48 -44.48 43.75 -43.75
cuboid  2 9 47.48 -47.48 47.48 -47.48 46.75 -46.75
cuboid  2 10 50.48 -50.48 50.48 -50.48 49.75 -49.75
cuboid  2 11 53.96 -53.96 53.96 -53.96 53.23 -53.23
end geom
read bias  id=400 2 11  end bias
end data
end
```

. . . . . .

```
#csas25
sample problem 6  one 2c8 unit (single unit)
27groupndf4                  infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 6  one sc8 unit (single unit)
read parm flx=yes fdn=yes far=yes end parm
```

```
read geometry
cylinder     1 1 5.748 5.3825 -5.3825
end geometry
end data
end
. . . . . .
#csas25
sample problem 7  bare 2c8 using specular reflection
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 7  bare 2c8 using specular reflection
read para  lng=40000 flx=yes fdn=yes far=yes end parameters
read geom  cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geom
read bounds +fc=specular end bounds  end data
end
. . . . . .
#csas25
sample problem 8  infinitely long cylinder from 2c8 unit
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 8  infinitely long cylinder from 2c8 unit
read param  lng=40000 end param
read geometry  cylinder  1 1 5.748 10.0 -10.0
cuboid  0 1 6.87 -6.87 6.87 -6.87 10.0 -10.0
end geometry
read bounds zfc=mirror  end bounds
end data
end
. . . . . .
#csas25
sample problem 9  infinite array of 2c8 units
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 9  infinite array of 2c8 units
read param  lng=40000 end param
end mixt
read boun  all=mir  end boun
```

```
read geom
cylinder  1 1 5.748 5.3825 -5.3825  cuboid  0 1 6.87 -6.87 6.87 -6.87
6.505 -6.505  end geom  end data
end
. . . . . .
#csas25
sample problem 10  case 2c8 bare  write restart
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-04 293 end
u-235  1  0.0 4.47971e-02 293 end
u-236  1  0.0 9.57231e-05 293 end
u-238  1  0.0 2.65767e-03 293 end
end comp
sample problem 10  case 2c8 bare  write restart
read parameters lng=40000 flx=yes fdn=yes far=yes res=5 wrs=94
end parameters
read geometry
cylinder 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geometry
read array  nux=2 nuy=2 nuz=2  end array
end data
end
. . . . . .
#csas25
sample problem 11  2c8 bare  read restart data
read param  lng=40000 beg=51  rst=94 res=0 end param
end data
end
. . . . . .
#csas25
sample problem 12 4 aqueous 4 metal mixed units
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-04 293 end
u-235  1  0.0 4.47971e-02 293 end
u-236  1  0.0 9.57231e-05 293 end
u-238  1  0.0 2.65767e-03 293 end
h      2  0.0 5.77964e-02 293 end
n      2  0.0 2.13092e-03 293 end
o      2  0.0 3.74130e-02 293 end
u-234  2  0.0 1.06784e-05 293 end
u-235  2  0.0 9.84599e-04 293 end
u-236  2  0.0 5.29385e-06 293 end
u-238  2  0.0 6.19413e-05 293 end
h      3  0.0 5.68187e-02 293 end
c      3  0.0 3.55117e-02 293 end
o      3  0.0 1.42047e-02 293 end
end comp
sample problem 12 4 aqueous 4 metal mixed units
read param  lng=40000 fdn=yes nub=yes smu=yes mkp=yes
mku=yes fmp=yes fmu=yes end param
```

```
read geom
box type 1
cylinder  2 1 9.525 8.89 -8.89
cylinder  3 1 10.16 9.525 -9.525
cuboid  0 1 10.875 -10.875 10.875 -10.875 10.24 -10.24
box type  2
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 6.59 -15.16 6.225 -14.255
box type  3
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 15.16 -6.59 6.225 -14.255
box type  4
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 6.59 -15.16 14.255 -6.225
box type  5
cylinder  1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.59 -15.16 15.16 -6.59 14.255 -6.225
end geom  read array  nux=2 nuy=2 nuz=2  loop
1 3r2 1 2 1 1 2 1  2 9r1  3 3r1 2 2 1 3r1  4 6r1 2 2 1 5 3r1 2 2 1 2 2 1
end array
end data
end
. . . . . .
#csas25
sample problem 13  two cuboids in a cylindrical annulus
27groupndf4                    infhommedium
u-234  1  0.0 4.80915e-4 293 end
u-235  1  0.0 4.46299e-2 293 end
u-236  1  0.0 9.53659e-5 293 end
u-238  1  0.0 2.64775e-3 293 end
end comp
sample problem 13
read param  lng=40000  end param
read geom
unit 1
cuboid  1 1 6.35 -6.35 6.35 -6.35 7.62 0.0
cylinder  0 1 13.97 7.62 0.0 orig -6.0934 0.0
cylinder  1 1 19.05 7.62 0.0 orig -6.0934 0.0
cuboid  0 1 12.9566 -25.1434 19.05 -19.05 7.62 0.0
unit 2
cuboid  1 1 6.35 -6.35 6.35 -6.35 8.56 0.0
cylinder  0 1 13.97 8.56 0.0 origin 6.0934 0.0
cylinder  1 1 19.05 8.56 0.0 origin 6.0934 0.0
cuboid  0 1 25.1434 -12.9566 19.05 -19.05 8.56 0.0
unit 3
cuboid  1 1 6.35 -6.35 6.35 -6.35 2.616 0.0
cuboid  0 1 25.1434 -12.9566 19.05 -19.05 2.616 0.0
end geom
read array nux=1 nuy=1 nuz=3  fill 1 2 3 t end array end data
end
. . . . . .
```

```
#csas25
sample problem 14  u metal cylinder in an annulus
27groupndf4                infhommedium
u-234  1  0.0 4.80915e-4 293 end
u-235  1  0.0 4.46299e-2 293 end
u-236  1  0.0 9.53659e-5 293 end
u-238  1  0.0 2.64775e-3 293 end
end comp
sample problem 14
read param lng=40000 end param
read geom
cylinder  1 1 8.89 10.109 0.0 orig 5.0799 0.0
cylinder  0 1 13.97 10.109 0.0
cylinder  1 1 19.05 10.109 0.0
end geom
end data
end
```

. . . . . .

```
#csas25
sample problem 15  small water reflected sphere on plexiglas collar
27groupndf4                infhommedium
u-234  1  0.0 5.65801e-04 293 end
u-235  1  0.0 4.70211e-02 293 end
u-236  1  0.0 9.58966e-05 293 end
u-238  1  0.0 4.65935e-04 293 end
h      2  0.0 5.68187e-02 293 end
c      2  0.0 3.55117e-02 293 end
o      2  0.0 1.42047e-02 293 end
h      3  0.0 6.67514e-02 293 end
o      3  0.0 3.33757e-02 293 end
end comp
sample problem 15  small water reflected sphere on plexiglas collar
read param  lng=40000  flx=yes fdn=yes end param
read geom
unit 1
hemisphe-z 1 1 6.5537 chord -5.09066
cylinder  3 1 4.1275 -5.09066 -7.63065
cylinder  2 1 12.7   -5.09066 -7.63065
cuboid  3 1 4p12.7 -5.09066 -7.63065
unit 2
hemisphe+z 1 1 6.5537 chord 5.09066
cuboid  3 1 4p12.7 6.5537 -5.09066
core  0 1 -12.7 -12.7 -7.092175
cylinder  3 1 17.97 2p7.0922
replicate 3 2 3*3.0 5
end geom
read bias  id=500 2 6 end bias
read array nux=1 nuy=1 nuz=2  fill 1 2 end array
read plot ttl='x-z slice through the center of the sphere' lpi=6
xul=-20.0 zul=10.0 yul=0.0  xlr=20.0 ylr=0.0 zlr=-10.0
uax=1.0 wdn=-1.0 nax=130    nch=' *0-'
```

```
end plot
end data
end
. . . . . .
#csas25
sample problem 16 uo2f2 infinite slab k-infinity
27groupndf4                    infhommedium
f      1  0.0 2.96286e-03 293 end
h      1  0.0 6.04824e-02 293 end
o      1  0.0 3.32041e-02 293 end
u-235  1  0.0 1.38188e-03 293 end
u-238  1  0.0 9.95503e-05 293 end
na     2  0.0 2.39502e-03 293 end
al     2  0.0 4.97719e-04 293 end
si     2  0.0 1.80267e-02 293 end
b-10   2  0.0 9.08241e-04 293 end
b-11   2  0.0 3.68719e-03 293 end
o      2  0.0 4.49173e-02 293 end
f      3  0.0 2.96286e-03 293 end
h      3  0.0 6.04824e-02 293 end
o      3  0.0 3.32041e-02 293 end
u-235  3  0.0 1.38188e-03 293 end
u-238  3  0.0 9.95503e-05 293 end
b-10   3  0.0 2.92803e-04 293 end
b-11   3  0.0 1.18870e-03 293 end
end comp
sample problem 16 uo2f2 infinite slab k-infinity
read parameters   lng=40000 amx=yes xap=no        end parameters
read geometry
cuboid 1 1 2.479 -2.479 100 -100 100 -100
cuboid 2 1 3.749 -3.749 100 -100 100 -100
cuboid 3 1 17.479 -17.479 100 -100 100 -100
end geom
read bounds  all=mirror  end bounds    read array  end array
end data
end
. . . . . .
#csas25
sample problem 17 93% uo2f2 solution sphere forward calculation
27groupndf4                    infhommedium
h      1  0.0 6.54785e-02 293 end
o      1  0.0 3.34202e-02 293 end
f      1  0.0 6.80923e-04 293 end
u-235  1  0.0 3.16909e-04 293 end
u-238  1  0.0 2.35521e-05 293 end
end comp
sample problem 17 93% uo2f2 solution sphere forward calculation
read parameters   lng=40000 amx=yes xap=no end parameters
read geometry
sphere 1 1 16.0
end geom
```

```
end data
end
. . . . . .
#csas25
sample problem 18   1f27 demonstration of options problem
27groupndf4                infhommedium
h      1  0.0 5.77964e-02 293 end
n      1  0.0 2.13092e-03 293 end
o      1  0.0 3.74130e-02 293 end
u-234  1  0.0 1.06784e-05 293 end
u-235  1  0.0 9.84599e-04 293 end
u-236  1  0.0 5.29385e-06 293 end
u-238  1  0.0 6.19413e-05 293 end
h      2  0.0 5.68187e-02 293 end
c      2  0.0 3.55117e-02 293 end
o      2  0.0 1.42047e-02 293 end
c      3  0.0 3.97311e-02 293 end
h      3  0.0 8.26407e-02 293 end
o      4  0.0 3.33757e-11 293 end
h      4  0.0 6.67514e-11 293 end
o      5  0.0 3.3371e-2   293 end
h      5  0.0 6.6674e-2   293 end
end comp
sample problem 18   1f27 demonstration of options problem
read para npg=1500 gen=200 nsk=20 fdn=yes nub=yes
  mku=yes fmu=yes mkh=yes fmh=yes mka=yes fma=yes rnd=f12c09ed2195
  pwt=yes far=yes flx=yes amx=yes pgm=yes tme=120 tba=2
 end para
read geom unit 1 cylinder 1 1 9.52 8.7804 -8.7804
 cylinder 0 1 9.52 8.9896 -8.7804
 cylinder 2 1 10.16 9.6296 -9.4204
 cuboid 4 1 18.45 -18.45 18.45 -18.45 17.8946 -17.6854
 unit 2 array 1 3*0.0
 unit 3 array 2 3*0.0
 unit 4 array 3 3*0.0
 unit 5 array 4 3*0.0
 global
 unit 6 cuboid 4 1 55.3501 -55.3501 55.3501 -55.3501 53.3701 -53.3701
  hole 2 -55.35   -18.45   -17.79
  hole 3 -55.35   -18.45   -53.3701
  hole 4  18.4501 -18.45   -53.3701
  hole 5 -55.3501 -55.3501 -53.3701
 replicate 3 2 6*3 5  replicate 3 7 6*0.24 1
 cuboid 5 1 70.5901 -70.5901 70.5901 -70.5901 68.6101 -99.0901 end geom
 read bias  id=400 2 7  end bias
read array
 ara=1 nux=2 nuy=2 nuz=2 fill f1 end fill
 ara=2 nux=2 nuy=2 nuz=1 fill f1 end fill
 ara=3 nux=1 nuy=2 nuz=3 fill f1 end fill
 ara=4 nux=3 nuy=1 nuz=3 fill f1 end fill
 end array
```

```
 read start nst=6 tfx=0.0 tfy=0.0 tfz=0.0
 lnu=1500 ps6=yes
 end start
read plot          plt=yes lpi=6
ttl=?  1f27 xy plot at z=0.0 ?  xul=-71.0 yul=71.0 zul=0.0
 xlr=71.0 ylr=-71.0 zlr=0.0 uax=1 vdn=-1 nax=130 nch=?.*-3 ?
 run=yes end       ttl=?unit map 1f27 xy plot at z=0.0?
 pic=unit nch=? 123456? end plot    end data
end
. . . . . .
#csas25
sample problem 19 4 aqueous 4 metal array of arrays (samp prob 12)
27groupndf4                infhommedium
u-234  1  0.0 4.82716e-04 293 end
u-235  1  0.0 4.47971e-02 293 end
u-236  1  0.0 9.57231e-05 293 end
u-238  1  0.0 2.65767e-03 293 end
h      2  0.0 5.77964e-02 293 end
n      2  0.0 2.13092e-03 293 end
o      2  0.0 3.74130e-02 293 end
u-234  2  0.0 1.06784e-05 293 end
u-235  2  0.0 9.84599e-04 293 end
u-236  2  0.0 5.29385e-06 293 end
u-238  2  0.0 6.19413e-05 293 end
h      3  0.0 5.68187e-02 293 end
c      3  0.0 3.55117e-02 293 end
o      3  0.0 1.42047e-02 293 end
end comp
sample problem 19 4 aqueous 4 metal array of arrays (samp prob 12)
read param lng=40000 flx=yes fdn=yes nub=yes smu=yes mkp=yes
mku=yes fmp=yes fmu=yes end param
read geom
unit 1
com='uranyl nitrate solution in a plexiglas container'
cylinder  2 1 9.525 2p8.89
cylinder  3 1 10.16 2p9.525
cuboid  0 1 4p10.875 2p10.24
unit 2
com='uranium metal cylinder'
cylinder  1 1 5.748 2p5.3825
cuboid  0 1 4p6.59 2p6.225
unit 3
com='1x2x2 array of solution units'
array 1 3*0.0
unit 4
com='1x2x2 array of metal units padded to match solution array'
array 2 3*0.0
replicate 0 1 2*0.0 2*8.57 2*8.03 1
end geom
read array ara=1 nux=1 nuy=2 nuz=2 fill f1 end fill
ara=2 nux=1 nuy=2 nuz=2 fill f2 end fill gbl=3 ara=3 nux=2 nuy=1 nuz=1
```

```
com='composite array of solution and metal units'
fill 4 3 end fill
end array
end data
end
. . . . . .
#csas25
sample problem 20 triangular pitched array
27groupndf4                    infhommedium
u-235  1  0.0 1.37751e-03 293 end
u-238  1  0.0 9.92354e-05 293 end
o      1  0.0 3.32049e-02 293 end
f      1  0.0 2.95349e-03 293 end
h      1  0.0 6.05028e-02 293 end
al     2  0.0 6.02374e-02 293 end
end comp
sample problem 20 triangular pitched array
read geom
unit 1
cylinder   1 1 10.16 18.288 0
cylinder   2 1 10.312 18.288 -.152
unit 2
cuboid     0 1 4p50 50 -.152
hole       1 3r0
hole       1 21.006 2r0
hole       1 -21.006 2r0
hole       1 10.503 18.192 0
hole       1 -10.503 18.192 0
hole       1 10.503 -18.192 0
hole       1 -10.503 -18.192 0
end geom
read array nux=1 nuy=1 nuz=1 fill 2 end fill end array
read plot ttl='hex array' pic=mix lpi=6
xul=0 yul=100 zul=10
xlr=100 ylr=0 zlr=10 uax=1 vdn=-1 nax=120
nch=' 12' end plot
end data
end
. . . . . .
#csas25
sample problem 21  partially filled sphere
27groupndf4                    infhommedium
h      1  0.0  6.15670e-02 293 end
o      1  0.0  3.32845e-02 293 end
f      1  0.0  2.50098e-03 293 end
u-234  1  0.0  2.54223e-07 293 end
u-235  1  0.0  6.18922e-05 293 end
u-238  1  0.0  1.18834e-03 293 end
al     2  0.0  6.02374e-02 293 end
end comp
sample problem 21
```

```
read param  lng=40000 lib=4  end param
read geom
hemisphe-z  1 1 34.6   chord 30.
sphere      0 1 34.6
sphere      2 1 34.759
end geom
end data
end
. . . . . .
#csas25
sample problem 22   case 2c8 bare with 3 nested, equal volume holes
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 22
read parameters  lng=40000 flx=yes fdn=yes far=yes end parameters
read geometry
unit 1
cylinder 1 1 3.621 2p3.3907
unit 2
cylinder 1 1 4.5622 2p4.2721
hole 1 3*0.0
unit 3
cylinder 1 1 5.2224 2p4.8903
hole 2 3*0.0
unit 4
cylinder 1 1 5.748 5.3825 -5.3825
hole 3 3*0.0
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geometry
read array  nux=2 nuy=2 nuz=2 fill f4 end fill end array
end data
end
. . . . . .
#csas25
sample problem 23  case 2c8 bare as mixed zhemicylinders
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 23
read parameters  lng=40000 fdn=yes end parameters
read geometry
unit 1
com='-x half of unit 3'
zhemicyl-x 1 1 5.748 5.3825 -5.3825
```

```
cuboid  0 1 0.0  -6.87 6.87 -6.87 6.505 -6.505
unit 2
com='+x half of unit 3'
zhemicyl+x 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87  0.0  6.87 -6.87 6.505 -6.505
unit 3
com='cylinder composed of equal halves (zhemicylinders with x radii)'
array 1 3*0.0
unit 4
com='-x portion (more than half) of unit 6'
zhemicyl-x 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 3.0  -6.87 6.87 -6.87 6.505 -6.505
unit 5
com='+x portion (less than half) of unit 6'
zhemicyl+x 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.87  3.0  6.87 -6.87 6.505 -6.505
unit 6
com='cylinder composed of unequal halves (zhemicylinders with x radii)'
array 2 3*0.0
unit 7
com='cylinder of a single zhemicylinder in the -x direction'
zhemicyl-x 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 8
com='cylinder of a single zhemicylinder in the +x direction'
zhemicyl+x 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 9
com='-y half of unit 11'
zhemicyl-y 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 0.0  -6.87 6.505 -6.505
unit 10
com='+y half of unit 11'
zhemicyl+y 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.87  0.0  6.505 -6.505
unit 11
com='cylinder composed of equal halves (zhemicylinders with z radii)'
array 3 3*0.0
unit 12
com='-y portion (more than half) of unit 14'
zhemicyl-y 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 6.87 -6.87 3.0  -6.87 6.505 -6.505
unit 13
com='+y portion (less than half) of unit 14'
zhemicyl+y 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.87 -6.87 6.87  3.0  6.505 -6.505
unit 14
com='cylinder composed of unequal halves (zhemicylinders with z radii)'
array 4 3*0.0
unit 15
com='cylinder of a single zhemicylinder in the -y direction'
```

```
zhemicyl-y 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
unit 16
com='cylinder of a single zhemicylinder in the +y'
zhemicyl+y 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.87 -6.87 6.505 -6.505
end geometry
read array
com='array 1 defines unit 3 (zhemicylinders with x radii)'
ara=1 nux=2 nuy=1 nuz=1 fill 1 2 end fill
com='array 2 defines unit 6 (zhemicylinders with x radii)'
ara=2 nux=2 nuy=1 nuz=1 fill 4 5 end fill
com='array 3 defines unit 11 (zhemicylinders with y radii)'
ara=3 nux=1 nuy=2 nuz=1 fill 9 10 end fill
com='array 4 defines unit 14 (zhemicylinders with y radii)'
ara=4 nux=1 nuy=2 nuz=1 fill 12 13 end fill
com='array 5 defines the total 2c8 problem'
gbl=5 ara=5 nux=2 nuy=2 nuz=2 fill 3 7 6 8 11 15 14 16 end fill
end array
end data
end
. . . . . .
#csas25
sample problem 24  case 2c8 bare as mixed xhemicylinders
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 24
read parameters  lng=40000 fdn=yes  end parameters
read geometry
unit 1
com='-y half of unit 3'
xhemicyl-y 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.505 -6.505 0.0 -6.87 6.87 -6.87
unit 2
com='+y half of unit 3'
xhemicyl+y 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.505 -6.505 6.87 0.0  6.87 -6.87
unit 3
com='cylinder composed of equal halves (xhemicylinders with y radii)'
array 1 3*0.0
unit 4
com='-y portion (more than half) of unit 6'
xhemicyl-y 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 6.505 -6.505 3.0 -6.87 6.87 -6.87
unit 5
com='+y portion (less than half) of unit 6'
xhemicyl+y 1 1 5.748 5.3825 -5.3825 chord -3.0
```

```
cuboid  0 1 6.505 -6.505 6.87 3.0  6.87 -6.87
unit 6
com='cylinder composed of unequal halves (xhemicylinders with y radii)'
array 2 3*0.0
unit 7
com='cylinder of a single xhemicylinder in the -y direction'
xhemicyl-y 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 -6.87
unit 8
com='cylinder of a single xhemicylinder in the +y direction'
xhemicyl+y 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 -6.87
unit 9
com='-z half of unit 11'
xhemicyl-z 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.505 -6.505 6.87 -6.87 0.0 -6.87
unit 10
com='+z half of unit 11'
xhemicyl+z 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 0.0
unit 11
com='cylinder composed of equal halves (xhemicylinders with z radii)'
array 3 3*0.0
unit 12
com='-z portion (more than half) of unit 14'
xhemicyl-z 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 6.505 -6.505 6.87 -6.87 3.0 -6.87
unit 13
com='+z portion (less than half) of unit 14'
xhemicyl+z 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 3.0
unit 14
com='cylinder composed of unequal halves (xhemicylinders with z radii)'
array 4 3*0.0
unit 15
com='cylinder of a single xhemicylinder in the -z direction'
xhemicyl-z 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 -6.87
unit 16
com='cylinder of a single xhemicylinder in the +z direction'
xhemicyl+z 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.505 -6.505 6.87 -6.87 6.87 -6.87
end geometry
read array
com='array 1 defines unit 3 (xhemicylinders with y radii)'
ara=1 nux=1 nuy=2 nuz=1 fill 1 2 end fill
com='array 2 defines unit 6 (xhemicylinders with y radii)'
ara=2 nux=1 nuy=2 nuz=1 fill 4 5 end fill
com='array 3 defines unit 11 (xhemicylinders with z radii)'
ara=3 nux=1 nuy=1 nuz=2 fill 9 10 end fill
com='array 4 defines unit 14 (xhemicylinders with z radii)'
```

```
ara=4 nux=1 nuy=1 nuz=2 fill 12 13 end fill
com='array 5 defines the total 2c8 problem'
gbl=5 ara=5 nux=2 nuy=2 nuz=2 fill 3 7 6 8 11 15 14 16 end fill
end array
end data
end
.....
#csas25
sample problem 25  case 2c8 bare as mixed yhemicylinders
27groupndf4                    infhommedium
u-234  1  0.0 4.82716e-4 293 end
u-235  1  0.0 4.47971e-2 293 end
u-236  1  0.0 9.57231e-5 293 end
u-238  1  0.0 2.65767e-3 293 end
end comp
sample problem 25
read parameters  lng=40000  fdn=yes end parameters
read geometry
unit 1
com='-x half of unit 3'
yhemicyl-x 1 1 5.748 5.3825 -5.3825
cuboid  0 1 0.0  -6.87 6.505 -6.505 6.87 -6.87
unit 2
com='+x half of unit 3'
yhemicyl+x 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87  0.0  6.505 -6.505 6.87 -6.87
unit 3
com='cylinder composed of equal halves (yhemicylinders with x radii)'
array 1 3*0.0
unit 4
com='-x portion (more than half) of unit 6'
yhemicyl-x 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 3.0  -6.87 6.505 -6.505 6.87 -6.87
unit 5
com='+x portion (less than half) of unit 6'
yhemicyl+x 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.87  3.0  6.505 -6.505 6.87 -6.87
unit 6
com='cylinder composed of unequal halves (yhemicylinders with x radii)'
array 2 3*0.0
unit 7
com='cylinder of a single yhemicylinder in the -x direction'
yhemicyl-x 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 -6.87
unit 8
com='cylinder of a single yhemicylinder in the +x direction'
yhemicyl+x 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 -6.87
unit 9
com='-z half of unit 11'
yhemicyl-z 1 1 5.748 5.3825 -5.3825
```

```
cuboid  0 1 6.87 -6.87 6.505 -6.505 0.0 -6.87
unit 10
com='+z half of unit 11'
yhemicyl+z 1 1 5.748 5.3825 -5.3825
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 0.0
unit 11
com='cylinder composed of equal halves (yhemicylinders with z radii)'
array 3 3*0.0
unit 12
com='-z portion (more than half) of unit 14'
yhemicyl-z 1 1 5.748 5.3825 -5.3825 chord 3.0
cuboid  0 1 6.87 -6.87 6.505 -6.505 3.0 -6.87
unit 13
com='+z portion (less than half) of unit 14'
yhemicyl+z 1 1 5.748 5.3825 -5.3825 chord -3.0
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 3.0
unit 14
com='cylinder composed of unequal halves (yhemicylinders with z radii)'
array 4 3*0.0
unit 15
com='cylinder of a single  yhemicylinder in the -z direction'
yhemicyl-z 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 -6.87
unit 16
com='cylinder of a single yhemicylinder in the +z direction'
yhemicyl+z 1 1 5.748 5.3825 -5.3825 chord 5.748
cuboid  0 1 6.87 -6.87 6.505 -6.505 6.87 -6.87
end geometry
read array
com='array 1 defines unit 3 (yhemicylinders with x radii)'
ara=1 nux=2 nuy=1 nuz=1 fill 1 2 end fill
com='array 2 defines unit 6 (yhemicylinders with x radii)'
ara=2 nux=2 nuy=1 nuz=1 fill 4 5 end fill
com='array 3 defines unit 11 (yhemicylinders with z radii)'
ara=3 nux=1 nuy=1 nuz=2 fill 9 10 end fill
com='array 4 defines unit 14 (zhemicylinders with z radii)'
ara=4 nux=1 nuy=1 nuz=2 fill 12 13 end fill
com='array 5 defines the total 2c8 problem'
gbl=5 ara=5 nux=2 nuy=2 nuz=2 fill 3 7 6 8 11 15 14 16 end fill
end array
end data
end
```

KENO V.a Input Listing for

MCNP Benchmark Problems

```
. . . . . .
#csas25
(mcnp primer ex.1) plutonium sphere with nickel coating
27groupndf4                infhommedium
pu-239  1  0.0  3.7047e-2 293 end
pu-240  1  0.0  1.7510e-3 293 end
pu-241  1  0.0  1.1700e-4 293 end
ni      2  0.0  9.1322e-2 293 end
end comp
plutonium sphere with nickel coating (mcnp primer ex.1)
read para npg=1500 end para
read geometry
sphere     1 1 6.38493
sphere     2 1 6.39763
end geometry
end data
end
. . . . . .
#csas25
(mcnp primer ex.2.1) bare plutonium cylinder
27groupndf4                infhommedium
pu-239  1  0.0  3.9802e-2 293 end
end comp
bare plutonium cylinder (mcnp primer ex.2.1)
read param npg=1500 end param
read geometry
cylinder   1 1 4.935 17.273 0.
end geometry
end data
end
. . . . . .
#csas25
(mcnp primer ex.2.2) plutonium cylinder reflected with nat. uranium
27groupndf4                infhommedium
pu-239  1  0.0  3.9802e-2 293 end
u-235   2  0.0  3.4246e-4 293 end
u-238   2  0.0  4.7219e-2 293 end
end comp
plutonium cylinder reflected with nat. uranium (mcnp primer ex.2.2)
read param npg=1500 end param
read geometry
cylinder   1 1 4.935 6.909 0.
cylinder   2 1 9.935 6.909 0.
end geometry
end data
end
. . . . . .
```

```
#csas25
(mcnp primer ex.2.3) plutonium cylinder reflected with nat. uranium
27groupndf4                infhommedium
pu-239  1  0.0  3.9802e-2 293 end
u-235   2  0.0  3.4246e-4 293 end
u-238   2  0.0  4.7219e-2 293 end
end comp
plutonium cylinder reflected with nat. uranium (mcnp primer ex.2.3)
read param npg=1500 end param
read geometry
cylinder   1 1 4.935 6.909 0.
cylinder   2 1 9.935 11.909 -5.0
end geometry
end data
end
```

. . . . . .

```
#csas25
(mcnp primer ex3) s(a,b) treatment mcnp primer
27groupndf4                infhommedium
h       1  0.0  5.7058e-2 293 end
o       1  0.0  3.2929e-2 293 end
f       1  0.0  4.3996e-3 293 end
u-235   1  0.0  1.0889e-4 293 end
u-238   1  0.0  2.0909e-3 293 end
al      2  0.0  6.02374e-2 293 end
end comp
example problem 3. s(a,b) treatment mcnp primer
read param flx=yes end parm
read geom
cylinder    1 1 20.12    100.0 0.0
cylinder    0 1 20.12    110.0 0.0
cylinder    2 1 20.2787 110.0 -0.1587
end geom
end data
end
```

. . . . . .

```
#csas25
(mcnp primer ex. 4) 2 uo2f2 cylinders in tank of water
27groupndf4                infhommedium
u-235   1  0.0  1.1760e-3 293 end
u-238   1  0.0  8.2051e-5 293 end
o       1  0.0  3.3621e-2 293 end
h       1  0.0  6.221e-2  293 end
f       1  0.0  2.5161e-3 293 end
al      2  0.0  6.0263e-2 293 end
h       3  0.0  6.6857e-2 293 end
o       3  0.0  3.3428e-2 293 end
end comp
2 uo2f2 cylinders in tank of water (mcnp primer ex. 4)
read geometry
```

```
unit 1
cylinder   1 1  6.35  70.2  0.0
cylinder   2 1  6.50  80.0 -0.15
cuboid     3 1  8.5  -16.5  26.5 -26.5 80.0 -20.15
unit 2
cylinder   1 1  6.35  70.2  0.0
cylinder   2 1  6.50  80.0 -0.15
cuboid     3 1  26.5 -8.5 26.5 -26.5 80.0 -20.15
global unit 3
array      1   -30.0 -26.0 -50.075
end geometry
read array nux=2 nuy=1 nuz=1
fill 1 2 end fill
end array
end data
end
. . . . . .
#csas25
(mcnp primer ex. 5) 3x2 array of pu(no3)4 cylinders
27groupndf4              infhommedium
pu-239  1  0.0  2.7682e-4 293 end
pu-240  1  0.0  1.2214e-5 293 end
pu-241  1  0.0  8.3390e-7 293 end
pu-242  1  0.0  4.5800e-8 293 end
h       1  0.0  6.0070e-2 293 end
o       1  0.0  3.6540e-2 293 end
n       1  0.0  2.3699e-3 293 end
fe      2  0.0  6.3310e-2 293 end
cr      2  0.0  1.6540e-2 293 end
ni      2  0.0  6.5100e-3 293 end
end comp
3x2 array of pu(no3)4 cylinders (mcnp primer ex. 5)
read para npg=1500 end para
read geometry
unit 1
cylinder   1 1  12.49 39.24  0.0
cylinder   0 1  12.49 101.7  0.0
cylinder   2 1  12.79 102.7 -1.0
cuboid     0 1  17.79 -17.79 17.79 -17.79 102.7 -1.0
global unit 2
array      1   -53.37 -35.58 -51.85
end geometry
read array nux=3 nuy=2 nuz=1
fill 1 1 1 1 1 1 end fill
end array
end data
end
. . . . . .
#csas25
p12:  jezebel at 95.5% pu-239
27groupndf4              infhommedium
```

```
pu-239  1  0.0  3.7555e-2 293 end
pu-240  1  0.0  1.7622e-3 293 end
end comp
jezebel at 95.5% pu-239
read para npg=1500 end para
read geometry
sphere     1 1 6.385
end geometry
end data
end
. . . . . .
#csas25
p13:  jezebel at 80% pu-239
27groupndf4                infhommedium
pu-239  1  0.0  3.1701e-2 293 end
pu-240  1  0.0  7.8922e-3 293 end
end comp
jezebel at 80% pu-239
read geometry
sphere     1 1 6.660
end geometry
end data
end
. . . . . .
#csas25
p21:  uranium cylinder with 10.9% u-235
27groupndf4                infhommedium
u-235  1  0.0  5.2028e-3 293 end
u-238  1  0.0  4.1993e-2 293 end
end comp
uranium cylinder with 10.9% u-235
read geometry
cylinder     1 1 26.65 119.392 0.
end geometry
end data
end
. . . . . .
#csas25
p22:  uranium cylinder with 14.11% u-235
27groupndf4                infhommedium
u-235  1  0.0  6.6555e-3 293 end
u-238  1  0.0  4.0002e-2 293 end
end comp
uranium cylinder with 14.11% u-235
read geometry
cylinder     1 1 26.65 44.239 0.
end geometry
end data
end
. . . . . .
#csas25
```

```
p31:  graphite reflected uranium sphere 93.3% u-235
27groupndf4                infhommedium
u-235  1  0.0  4.5037e-2 293 end
u-238  1  0.0  3.0914e-3 293 end
c      2  0.0  8.3389e-2 293 end
fe     2  0.0  6.1227e-5 293 end
s      2  0.0  5.0181e-5 293 end
end comp
graphite reflected uranium sphere 93.3% u-235
read para npg=1500 end para
read geometry
sphere     1 1 7.39840
sphere     2 1 12.4984
end geometry
end data
end
. . . . . .
#csas25
p32:  water reflected uranium sphere 97.67% u-235
27groupndf4                infhommedium
u-234  1  0.0  5.3000e-4 293 end
u-235  1  0.0  4.7030e-2 293 end
u-236  1  0.0  1.0000e-4 293 end
u-238  1  0.0  4.9000e-4 293 end
h      2  0.0  6.6790e-2 293 end
o      2  0.0  3.3400e-2 293 end
end comp
water reflected uranium sphere 97.67% u-235
read geometry
sphere     1 1 6.5537
cylinder   2 1 30.0 35.0 -35.0
end geometry
end data
end
. . . . . .
#csas25
p41:  3 uo2f2 cylinders in triangle configuration
27groupndf4                infhommedium
u-235  1  0.0  2.1936e-4 293 end
u-238  1  0.0  1.5736e-5 293 end
h      1  0.0  6.7709e-2 293 end
o      1  0.0  3.4291e-2 293 end
f      1  0.0  4.6964e-4 293 end
al     2  0.0  6.0486e-2 293 end
end comp
3 uo2f2 cylinders in triangle configuration
read para npg=1500 end para
read geometry
unit 1
cylinder   1 1  10.15  20.7  -20.7
cylinder   2 1  10.30  20.85 -20.85
```

```
global unit 2
cuboid     0 1  20.98 -20.98  30.0 -10.30 20.85 -20.85
hole       1    10.49   0.    0.
hole       1   -10.49   0.    0.
hole       1    0.     18.169 0.
end geometry
end data
end
```

KENO V.a Input Listing for

Additional Benchmark Problems

```
. . . . . .
#csas25
prob.1 water reflected pu metal sphere
27groupndf4                infhommedium
pu-239  1  0.0  4.8973e-2 293 end
h       2  0.0  6.6737e-2 293 end
o       2  0.0  3.3368e-2 293 end
end comp
prob.1 water reflected pu metal sphere
read geometry
sphere     1 1 4.11
sphere     2 1 24.11
end geometry
end data
end
. . . . . .
#csas25
prob.2 water reflected pu nitrate sphere
27groupndf4                infhommedium
pu-239  1  0.0  3.3662e-4 293 end
pu-240  1  0.0  1.6163e-5 293 end
n       1  0.0  2.7595e-3 293 end
h       1  0.0  6.0260e-2 293 end
o       1  0.0  3.7734e-2 293 end
fe      2  0.0  5.8886e-2 293 end
cr      2  0.0  1.7672e-2 293 end
ni      2  0.0  8.2374e-3 293 end
h       3  0.0  6.6063e-2 293 end
o       3  0.0  3.3032e-2 293 end
end comp
prob.2 water reflected pu nitrate sphere
read geometry
sphere     1 1 14.568
sphere     2 1 14.692
sphere     3 1 39.6922
end geometry
end data
end
. . . . . .
#csas25
prob.3 bare reflected pu nitrate sphere
27groupndf4                infhommedium
pu-239  1  0.0  2.3201e-5 293 end
pu-240  1  0.0  6.0140e-7 293 end
pu-241  1  0.0  1.7720e-8 293 end
n       1  0.0  7.6043e-4 293 end
o       1  0.0  3.4367e-2 293 end
```

```
h       1  0.0  6.4836e-2 293 end
al      2  0.0  6.0260e-2 293 end
end comp
prob.3 bare reflected pu nitrate sphere
read geometry
sphere     1 1 61.01
sphere     2 1 61.79
end geometry
end data
end
. . . . . .
#csas25
prob.4 bare u-233 nitrate solution in a sphere
27groupndf4                infhommedium
u-233   1  0.0  4.3293e-5 293 end
u-234   1  0.0  7.1479e-7 293 end
u-235   1  0.0  1.7574e-8 293 end
u-238   1  0.0  2.7763e-7 293 end
o       1  0.0  3.3721e-2 293 end
n       1  0.0  8.8300e-5 293 end
h       1  0.0  6.6747e-2 293 end
al      2  0.0  6.0263e-2 293 end
end comp
prob.4 bare u-233 nitrate solution in a sphere
read geometry
sphere     1 1 34.595
sphere     2 1 34.9252
end geometry
end data
end
. . . . . .
#csas25
prob.5 godiva
27groupndf4                infhommedium
u-234  1  0.0  4.8271e-4 293 end
u-235  1  0.0  4.4796e-2 293 end
u-236  1  0.0  9.6028e-5 293 end
u-238  1  0.0  2.6571e-3 293 end
end comp
prob.5 godiva
read param flx=yes fdn=yes far=yes end param
read geometry
sphere     1 1 8.72
end geometry
end data
end
. . . . . .
#csas25
prob.6 bare u-235 nitrate solution in a sphere
27groupndf4                infhommedium
u-234   1  0.0  5.3800e-7 293 end
```

```
u-235   1  0.0  4.8066e-5 293 end
u-236   1  0.0  1.3800e-7 293 end
u-238   1  0.0  2.8070e-6 293 end
o       1  0.0  3.3687e-2 293 end
n       1  0.0  1.8690e-4 293 end
h       1  0.0  6.6228e-2 293 end
al      2  0.0  6.0263e-2 293 end
end comp
prob.6 bare u-235 nitrate solution in a sphere
read geometry
sphere     1 1 34.595
sphere     2 1 34.9252
end geometry
end data
end
. . . . . .
#csas25
prob.7 low enriched unmoderated mox sphere
27groupndf4                infhommedium
pu-239  1  0.0  1.9591e-3 293 end
u-235   1  0.0  1.6460e-4 293 end
u-238   1  0.0  2.2450e-2 293 end
o       1  0.0  4.9148e-2 293 end
h       2  0.0  6.6680e-2 293 end
o       2  0.0  3.3343e-2 293 end
end comp
prob.7 low enriched unmoderated mox sphere
read geometry
sphere     1 1 19.4
sphere     2 1 39.4
end geometry
end data
end
. . . . . .
#csas25
prob.8 water reflecte pu-u nitrate in cyl.
27groupndf4                infhommedium
pu-238  1  0.0  8.3767e-9 293 end
pu-239  1  0.0  2.9365e-5 293 end
pu-240  1  0.0  1.7386e-6 293 end
pu-241  1  0.0  1.0752e-7 293 end
pu-242  1  0.0  2.1964e-8 293 end
u-235   1  0.0  5.0609e-7 293 end
u-236   1  0.0  1.4077e-8 293 end
u-238   1  0.0  7.5163e-5 293 end
h       1  0.0  6.3896e-2 293 end
o       1  0.0  3.4957e-2 293 end
n       1  0.0  1.0883e-3 293 end
fe      2  0.0  6.3310e-2 293 end
cr      2  0.0  1.6540e-2 293 end
ni      2  0.0  6.5100e-3 293 end
```

```
h       3  0.0  6.6680e-2 293 end
o       3  0.0  3.3343e-2 293 end
end comp
prob.8 water reflecte pu-u nitrate in cyl.
read geometry
cylinder   1 1 30.48   95.20 0.0
cylinder   0 1 30.48  106.6  0.0
cylinder   2 1 30.559 106.6 -0.95
cylinder   3 1 45.559 106.6 -20.95
end geometry
end data
end
. . . . . .
#csas25
prob.9 water reflecte pu-u-gd nitrate in cyl.
27groupndf4                 infhommedium
pu-238  1  0.0  2.3500e-6 293 end
pu-239  1  0.0  1.8260e-4 293 end
pu-240  1  0.0  1.1180e-5 293 end
pu-241  1  0.0  7.1600e-7 293 end
pu-242  1  0.0  1.1900e-7 293 end
u-234   1  0.0  2.7000e-8 293 end
u-235   1  0.0  3.0130e-6 293 end
u-236   1  0.0  5.4000e-8 293 end
u-238   1  0.0  4.4420e-4 293 end
h       1  0.0  5.5178e-2 293 end
o       1  0.0  3.8688e-2 293 end
n       1  0.0  3.7278e-3 293 end
gd      1  0.0  4.0600e-6 293 end
fe      2  0.0  6.3310e-2 293 end
cr      2  0.0  1.6540e-2 293 end
ni      2  0.0  6.5100e-3 293 end
h       3  0.0  6.6680e-2 293 end
o       3  0.0  3.3343e-2 293 end
end comp
prob.9 water reflecte pu-u-gd nitrate in cyl.
read geometry
cylinder   1 1 30.154  75.3  0.0
cylinder   0 1 30.154 106.6  0.0
cylinder   2 1 30.593 106.6 -0.95
cylinder   3 1 45.593 106.6 -20.95
end geometry
end data
end
. . . . . .
#csas25
prob.11 4x4x4 array of pu metal cylinders
27groupndf4                 infhommedium
pu-239  1  0.0  4.6053e-2 293 end
pu-240  1  0.0  2.9263e-3 293 end
pu-241  1  0.0  2.2454e-4 293 end
```

```
pu-242  1  0.0  4.8612e-6 293 end
end comp
prob.11 4x4x4 array of pu metal cylinders
read para npg=1500 end para
read geometry
unit 1
cylinder   1 1 3.2625 2.3165 -2.3165
cuboid     0 1 6.255 -6.255   6.255 -6.255  3.93 -3.93
global unit 2
array      1  -25.02 -25.02 -15.72
end geometry
read array
ara=1 nux=4 nuy=4 nuz=4
fill 64r1 end fill
end array
end data
end
. . . . . .
#csas25
sheba2
27groupndf4                infhommedium
u-238  1  0.0  2.4990e-3 293 end
u-235  1  0.0  1.3190e-4 293 end
u-234  1  0.0  6.8450e-7 293 end
u-236  1  0.0  1.3160e-6 293 end
o      1  0.0  3.2100e-2 293 end
f      1  0.0  5.3340e-3 293 end
h      1  0.0  5.3540e-2 293 end
fe     2  0.0  5.7107e-2 293 end
cr     2  0.0  1.6768e-2 293 end
ni     2  0.0  7.4279e-3 293 end
mn     2  0.0  1.6706e-3 293 end
end comp
sheba2
read geometry
unit 1
cylinder   0 1  2.36 42.85 0.0
cylinder   2 1  3.0  42.85 0.0
cylinder   1 1  23.9 42.85 0.0
cylinder   2 1  25.4 42.85 0.0
cuboid     0 1  25.4 -25.4 25.4 -25.4 42.85 0.0
unit 2
cylinder   0 1  2.36 73.06 42.85
cylinder   2 1  3.0  73.06 42.85
cylinder   0 1  23.9 73.06 42.85
cylinder   2 1  25.4 73.06 42.85
cuboid     0 1  25.4 -25.4 25.4 -25.4 73.06 42.85
unit 3
cylinder   0 1  2.36 0.0 -1.9
cylinder   2 1  3.0  0.0 -2.54
cylinder   2 1  25.4 0.0 -2.54
```

```
cuboid     0 1  25.4 -25.4 25.4 -25.4 0.0 -2.54
unit 4
cylinder   2 1  25.4 75.6 73.06
cuboid     0 1  25.4 -25.4 25.4 -25.4 75.6 73.06
global unit 5
array      1    -25.4 -25.4 -2.54
end geometry
read array
ara=1  nux=1 nuy=1 nuz=4
fill 3 1 2 4 end fill
end array
end data
end
```

. . . . . .

Appendix E

MCNP Input Listing for Benchmark Problems

MCNP Input Listing

for 25 Problem KENO Test Set[27]

```
e5mt.1:  converted from keno file k.1; continuos energy;  endf/b-5
c     8 bare cylinders of U-metal
c     cell cards
c
1    1   4.80368e-2  -7  -8  9   imp:n=1  u=-1
2    0         #1               imp:n=1  u=1
3    0          -1  2  -3  4  -5  6   imp:n=1  u=2  lat=1
     fill=0:1  0:1  0:1  1  1  1  1  1  1  1  1
4    0     -11 12 -13 14 -15 16    imp:n=1  fill=2
5    0      #4  imp:n=0
c
c    surface cards
c
c   parallelpiped

1    px 0.0
2    px -13.74
3    py  0.0
4    py  -13.74
5    pz  0.0
6    pz  -13.01
c   cylinder
7    c/z  -6.87 -6.87  5.748
8    pz  -1.1225
9    pz  -11.8875
c   parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   px 13.7399
12   px -13.7399
13   py 13.7399
14   py -13.7399
15   pz 13.0099
16   pz -13.0099
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
```

```
c   default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1          $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 400 0
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=-6.87  -6.87  -6.505   axs=0 0 1
c
si1   l  4:3(1 1 0):1    $ path:  /cell4/cell3/lattice(1,1,0)/cell1
         4:3(1 0 0):1    $ etc.
         4:3(0 1 0):1    $ this ordering chosen to match
         4:3(0 0 0):1    $  sampling  in e5ce.2
         4:3(1 1 1):1    $
         4:3(1 0 1):1    $
         4:3(0 1 1):1    $
         4:3(0 0 1):1    $
sp1   1 1 1 1 1 1 1 1    $ equal probability for all paths above
c
sp2  -3                 $ Watt fission spectrum
c
si3    0.0    5.748      $ radial distribution
sp3  -21  1             $ p(x) = const*abs(x)
c
si4   -5.3825  5.3825    $ axial distribution
sp4   -21  0            $ p(x)= const
c
prdmp  j  j  1  j        $ write mctal file
c
print                   $ full output
. . . . . .
e5mt.2:  converted from keno file k.1; continuos energy;  endf/b-5
c     8 bare cylinders of U-metal
c     cell cards
c
1    1   4.80368e-2  -7  -8  9   imp:n=1  u=-1
2    0         #1              imp:n=1  u=1
3    0          -1  2  -3  4  -5  6   imp:n=1  u=2  lat=1
     fill=0:1  0:1  0:1  1  1  1  1  1  1  1  1
4    0     -11 12 -13 14 -15 16    imp:n=1  fill=2
5    0      #4  imp:n=0
c
c    surface cards
c
c   parallelpiped

1    px 0.0
2    px -13.74
```

```
3    py  0.0
4    py  -13.74
5    pz  0.0
6    pz  -13.01
c   cylinder
7    c/z  -6.87 -6.87  5.748
8    pz  -1.1225
9    pz  -11.8875
c   parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   px 13.7399
12   px -13.7399
13   py 13.7399
14   py -13.7399
15   pz 13.0099
16   pz -13.0099
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 400 0
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=-6.87  -6.87  -6.505   axs=0 0 1
c
si1   l  4:3(1 1 0):1    $ path:  /cell4/cell3/lattice(1,1,0)/cell1
         4:3(1 0 0):1    $ etc.
         4:3(0 1 0):1    $ this ordering chosen to match
         4:3(0 0 0):1    $  sampling  in e5ce.2
         4:3(1 1 1):1    $
         4:3(1 0 1):1    $
         4:3(0 1 1):1    $
         4:3(0 0 1):1    $
sp1   1 1 1 1 1 1 1 1    $ equal probability for all paths above
c
```

```
sp2  -3                $ Watt fission spectrum
c
si3   0.0   5.748      $ radial distribution
sp3  -21  1            $ p(x) = const*abs(x)
c
si4   -5.3825  5.3825   $ axial distribution
sp4   -21  0           $ p(x)= const
c
prdmp  j  j  1  j      $ write mctal file
c
print                  $ full output
```

. . . . . .

```
e5mt.1:  converted from keno file k.3; continuos energy;  endf/b-5
c   universes of paraffin with constant importance surrounding core
c
c    cell cards
c
c    lattice with cylinders of u fuel
1   1   4.80368e-2  -7  -8  9        imp:n=1  u=1       $ u cylinder
2   0         #1                  imp:n=1  u=1       $ outside
3   0          -1  2  -3  4  -5  6   imp:n=1  u=2  lat=1 $ 2x2x2 lattice
    fill=0:1  0:1  0:1  1  1  1  1  1  1  1  1          $ filling u's
4   2   0.122282 #3                 imp:n=1  u=2       $ outside
c
c   concentric boxes of constant importance
10  0        -11 12 -13 14 -15 16    imp:n=1  u=10  fill=2
11  2   0.122282 #10                imp:n=1  u=10
20  0        -21 22 -23 24 -25 26    imp:n=1  u=20  fill=10
21  2   0.122282 #20                imp:n=1  u=20
30  0        -31 32 -33 34 -35 36    imp:n=1  u=30  fill=20
31  2   0.122282 #30                imp:n=1  u=30
40  0        -41 42 -43 44 -45 46    imp:n=1  u=40  fill=30
41  2   0.122282 #40                imp:n=1  u=40
50  0        -51 52 -53 54 -55 56    imp:n=1  u=50  fill=40
51  2   0.122282 #50                imp:n=1  u=50
60  0        -61 62 -63 64 -65 66    imp:n=1        fill=50
61  0          #60                 imp:n=0
c
c   surface cards
c

c  parallelpiped
1   px  0.0
2   px -23.48
3   py  0.0
4   py -23.48
5   pz  0.0
6   pz -22.75
c  cylinder
7   c/z -11.74 -11.74 5.748
8   pz  -5.9925
```

```
9    pz  -16.7575
c   parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   px  23.479
12   px -23.479
13   py  23.479
14   py -23.479
15   pz  22.749
16   pz -22.749
c   parallelpiped
21   px  26.48
22   px -26.48
23   py  26.48
24   py -26.48
25   pz  25.75
26   pz -25.75
c   parallelpiped
31   px  29.48
32   px -29.48
33   py  29.48
34   py -29.48
35   pz  28.75
36   pz -28.75
c   parallelpiped
41   px  32.48
42   px -32.48
43   py  32.48
44   py -32.48
45   pz  31.75
46   pz -31.75
c   parallelpiped
51   px  35.48
52   px -35.48
53   py  35.48
54   py -35.48
55   pz  34.75
56   pz -34.75
c   parallelpiped
61   px  38.72
62   px -38.72
63   py  38.72
64   py -38.72
65   pz  37.99
66   pz -37.99
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
```

```
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c  paraffin
m2   1001.50c 0.675324      $ h (in paraffin)
     6000.50c 0.324676      $ c (in paraffin)
c
c    S(alpha,beta)
mt2  poly.01t
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1          $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 4500 0
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=-11.74  -11.74  -11.375   axs=0 0 1
c
si1   l  60:50:40:30:20:10:3(0 0 0):1  $ path: /cell60/cell50..../cell1
         60:50:40:30:20:10:3(1 0 0):1  $ path
         60:50:40:30:20:10:3(1 1 0):1  $ etc
         60:50:40:30:20:10:3(0 1 0):1  $
         60:50:40:30:20:10:3(0 0 1):1  $
         60:50:40:30:20:10:3(1 0 1):1  $
         60:50:40:30:20:10:3(0 1 1):1  $
         60:50:40:30:20:10:3(1 1 1):1  $
sp1   1 1 1 1 1 1 1 1    $ equal probability for all paths above
c
sp2  -3                  $ Watt fission spectrum
c
si3    0.0    5.748      $ radial distribution
sp3  -21  1              $ p(x) = const*abs(x)
c
si4   -5.3825  5.3825     $ axial distribution
sp4   -21  0             $ p(x)= const
c
prdmp  j  j  1  j        $ write mctal file
c
print                    $ full output
. . . . . .
e5mt.4:  converted from keno file k.4; continuos energy;  endf/b-5
c    universes of paraffin with constant importance surrounding core
c
c     cell cards
c
c     lattice with cylinders of u fuel
1    1   4.80368e-2  -7  -8  9        imp:n=1  u=1        $ u cylinder
```

```
2   0        #1                  imp:n=1  u=1       $ outside
3   0         -1  2  -3  4  -5  6   imp:n=1  u=2  lat=1 $ 2x2x2 lattice
    fill=0:1  0:1  0:1  1  1  1  1  1  1  1  1          $ filling u's
4   2  0.122282 #3                imp:n=1  u=2       $ outside
c
c   concentric boxes of constant importance
10  0        -11 12 -13 14 -15 16   imp:n=1    u=10  fill=2
11  2  0.122282 #10               imp:n=0.1  u=10
20  0        -21 22 -23 24 -25 26   imp:n=1.0  u=20  fill=10
21  2  0.122282 #20               imp:n=0.1  u=20
30  0        -31 32 -33 34 -35 36   imp:n=1.0  u=30  fill=20
31  2  0.122282 #30               imp:n=0.05 u=30
40  0        -41 42 -43 44 -45 46   imp:n=1.0  u=40  fill=30
41  2  0.122282 #40               imp:n=0.05 u=40
50  0        -51 52 -53 54 -55 56   imp:n=1.0  u=50  fill=40
51  2  0.122282 #50               imp:n=0.01 u=50
60  0        -61 62 -63 64 -65 66   imp:n=1.0       fill=50
61  0          #60                imp:n=0
c
c   surface cards
c

c   parallelpiped
1    px  0.0
2    px -23.48
3    py  0.0
4    py -23.48
5    pz  0.0
6    pz -22.75
c   cylinder
7    c/z -11.74 -11.74 5.748
8    pz  -5.9925
9    pz  -16.7575
c   parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   px  23.479
12   px -23.479
13   py  23.479
14   py -23.479
15   pz  22.749
16   pz -22.749
c   parallelpiped
21   px  26.48
22   px -26.48
23   py  26.48
24   py -26.48
25   pz  25.75
26   pz -25.75
c   parallelpiped
31   px  29.48
32   px -29.48
33   py  29.48
```

```
34   py -29.48
35   pz  28.75
36   pz -28.75
c   parallelpiped
41   px  32.48
42   px -32.48
43   py  32.48
44   py -32.48
45   pz  31.75
46   pz -31.75
c   parallelpiped
51   px  35.48
52   px -35.48
53   py  35.48
54   py -35.48
55   pz  34.75
56   pz -34.75
c   parallelpiped
61   px  38.72
62   px -38.72
63   py  38.72
64   py -38.72
65   pz  37.99
66   pz -37.99
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c  paraffin
m2   1001.50c 0.675324      $ h (in paraffin)
     6000.50c 0.324676      $ c (in paraffin)
c
c    S(alpha,beta)
mt2  poly.01t
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 4500 0
```

```
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=-11.74  -11.74  -11.375   axs=0 0 1
c
si1   l  60:50:40:30:20:10:3(0 0 0):1  $ path: /cell60/cell50..../cell1
         60:50:40:30:20:10:3(1 0 0):1  $ path
         60:50:40:30:20:10:3(1 1 0):1  $ etc
         60:50:40:30:20:10:3(0 1 0):1  $
         60:50:40:30:20:10:3(0 0 1):1  $
         60:50:40:30:20:10:3(1 0 1):1  $
         60:50:40:30:20:10:3(0 1 1):1  $
         60:50:40:30:20:10:3(1 1 1):1  $
sp1   1 1 1 1 1 1 1 1    $ equal probability for all paths above
c
sp2  -3                  $ Watt fission spectrum
c
si3    0.0    5.748      $ radial distribution
sp3  -21  1               $ p(x) = const*abs(x)
c
si4   -5.3825  5.3825     $ axial distribution
sp4   -21  0              $ p(x)= const
c
prdmp  j  j  1  j         $ write mctal file
c
print                    $ full output
. . . . . .
e5mt.5:  converted from keno file k.5; continuos energy;  endf/b-5
c    30.48 cm of paraffin surrounding metal cylinders
c    models the keno albedo option
c
c     cell cards
c
c     lattice with cylinders of u fuel
1    1   4.80368e-2  -7  -8  9          imp:n=1  u=1        $ u cylinder
2    0          #1                    imp:n=1  u=1        $ outside
3    0           -1  2  -3  4  -5  6   imp:n=1  u=2  lat=1 $ 2x2x2 lattice
     fill=0:1  0:1  0:1  1  1  1  1  1  1  1  1            $ filling u's
4    2   0.122282 #3                  imp:n=1  u=2        $ outside
c
c    concentric boxes of constant importance
10   0         -11 12 -13 14 -15 16    imp:n=1    u=10  fill=2
11   2   0.122282 #10                 imp:n=0.1  u=10
20   0         -21 22 -23 24 -25 26    imp:n=1.0  u=20  fill=10
21   2   0.122282 #20                 imp:n=0.1  u=20
30   0         -31 32 -33 34 -35 36    imp:n=1.0  u=30  fill=20
31   2   0.122282 #30                 imp:n=0.05 u=30
40   0         -41 42 -43 44 -45 46    imp:n=1.0  u=40  fill=30
41   2   0.122282 #40                 imp:n=0.05 u=40
50   0         -51 52 -53 54 -55 56    imp:n=1.0  u=50  fill=40
51   2   0.122282 #50                 imp:n=0.01 u=50
60   0         -61 62 -63 64 -65 66    imp:n=1.0        fill=50
61   0           #60                 imp:n=0
```

```
c
c   surface cards
c

c  parallelpiped
1    px  0.0
2    px -23.48
3    py  0.0
4    py -23.48
5    pz  0.0
6    pz -22.75
c  cylinder
7    c/z -11.74 -11.74 5.748
8    pz  -5.9925
9    pz  -16.7575
c  parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   px  23.479
12   px -23.479
13   py  23.479
14   py -23.479
15   pz  22.749
16   pz -22.749
c  parallelpiped
21   px  26.48
22   px -26.48
23   py  26.48
24   py -26.48
25   pz  25.75
26   pz -25.75
c  parallelpiped
31   px  29.48
32   px -29.48
33   py  29.48
34   py -29.48
35   pz  28.75
36   pz -28.75
c  parallelpiped
41   px  32.48
42   px -32.48
43   py  32.48
44   py -32.48
45   pz  31.75
46   pz -31.75
c  parallelpiped
51   px  35.48
52   px -35.48
53   py  35.48
54   py -35.48
55   pz  34.75
56   pz -34.75
c  parallelpiped
```

```
61   px  53.48
62   px -53.48
63   py  53.48
64   py -53.48
65   pz  53.23
66   pz -53.23
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c  paraffin
m2   1001.50c 0.675324      $ h (in paraffin)
     6000.50c 0.324676      $ c (in paraffin)
c
c    S(alpha,beta)
mt2  poly.01t
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 4500 0
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=-11.74  -11.74  -11.375   axs=0 0 1
c
si1   l  60:50:40:30:20:10:3(0 0 0):1  $ path: /cell60/cell50..../cell1
         60:50:40:30:20:10:3(1 0 0):1  $ path to cell 1 thru lattice(1,0,0)
         60:50:40:30:20:10:3(1 1 0):1  $ path to cell 1 thru lattice(1,1,0)
         60:50:40:30:20:10:3(0 1 0):1  $ etc.
         60:50:40:30:20:10:3(0 0 1):1  $
         60:50:40:30:20:10:3(1 0 1):1  $
         60:50:40:30:20:10:3(0 1 1):1  $
         60:50:40:30:20:10:3(1 1 1):1  $
sp1   1 1 1 1 1 1 1 1    $ equal probability for all paths above
c
sp2  -3                  $ Watt fission spectrum
c
si3    0.0    5.748      $ radial distribution
sp3  -21  1               $ p(x) = const*abs(x)
c
```

```
si4   -5.3825  5.3825    $ axial distribution
sp4   -21  0             $ p(x)= const
c
prdmp  j  j  1  j         $ write mctal file
c
print                     $ full output
. . . . . .
em6mt.6  KENO V.a Sample Problem 6
c     cell cards
1     1    4.8033e-2  -1 -2 3  imp:n=1
2     0                1:2:-3  imp:n=0

c     surface cards
1     cz  5.748
2     pz  5.3825
3     pz  -5.3825

c     data cards
mode  n
m1    92234.50c    1.005e-2
      92235.50c    9.3263e-1
      92236.50c    1.9929e-3
      92238.50c    5.533e-2
kcode   300  0.7  3  103  400  0
sdef  cel=1
prdmp
print
. . . . . .
e5mt.7:  converted from keno file k.7; continuos energy;  endf/b-5
c     reflection on 3 sides
c
c     cell cards
c
1     1   4.80368e-2  -7  -8  9            imp:n=1
2     0           (-1  2  -3  4  -5  6) #1 imp:n=1
3     0           1: -2:  3: -4:  5: -6    imp:n=0
c

c     surface cards
c
c   parallelpiped
*1    px  6.87              $ reflecting surface
2     px -6.87
*3    py  6.87              $ reflecting surface
4     py -6.87
*5    pz  6.505             $ reflecting surface
6     pz -6.505
c   cylinder
7     cz  5.748
8     pz  5.3825
9     pz -5.3825
```

```
c   enclosing sphere
10   so  11.0
c

c  data cards
c
mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   0.025e-6  1.0e-6  1.0e-4 1.0e-2 1.0e-1 5.0e-1 1.0
     2.0  4.0  10.0  14.0   20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 4500 0
c
sdef  cel=1  pos=0 0 0 axs=0 0 1 rad=d1  ext=d2  erg=d3
c
si1   0.0 5.748          $ be sure this encloses cell 1
c
si2  -5.3825 5.3825      $ be sure this encloses cell 1
c
sp3  -3                 $ watt fission spectrum
c
prdmp  j  j  1  j        $ write mctal file
c
print                   $ full output
. . . . . .
e5mt.8:  converted from keno file k.8; continuos energy;  endf/b-5
c    reflection on z face only
c
c     cell cards
c
1    1   4.80368e-2  -7  -5  6            imp:n=1
2    0          (-1  2  -3  4  -5  6) 7   imp:n=1
3    0          1: -2:  3: -4:  5: -6     imp:n=0
c

c    surface cards
c
c   parallelpiped
```

```
1   px  6.87
2   px -6.87
3   py  6.87
4   py -6.87
*5  pz  10.00              $ reflecting surface
*6  pz -10.00              $ reflecting surface
c   cylinder
7    cz  5.748
c

c  data cards
c
mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   0.025e-6  1.0e-6  1.0e-4 1.0e-2 1.0e-1 5.0e-1 1.0
     2.0  4.0  10.0  14.0   20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 4500 0
c
sdef  cel=1  pos=0 0 0 axs=0 0 1 rad=d1  ext=d2  erg=d3
c
si1   0.0 5.748          $ be sure this encloses cell 1
c
si2  -10.0 10.0          $ be sure this encloses cell 1
c
sp3  -3                 $ watt fission spectrum
c
prdmp  j  j  1  j        $ write mctal file
c
print                   $ full output
. . . . . .
e5mt.9:  converted from keno file k.9; continuos energy;  endf/b-5
c     infinite array of cylinder-in-box units
c     model by making all box walls reflective
c
c   cell cards
c
1    1   4.80368e-2  -7  -8  9   imp:n=1  u=1
```

```
2   0        #1              imp:n=1  u=1
3   0         -1  2  -3  4  -5  6   imp:n=1 fill=1
4   0          1:-2   3:-4:  5:-6  imp:n=0

c
c  surface cards
c
c  parallelpiped
*1    px 0.0
*2    px -13.74
*3    py  0.0
*4    py -13.74
*5    pz  0.0
*6    pz -13.01
c  cylinder
7    c/z  -6.87 -6.87  5.748
8    pz  -1.1225
9    pz  -11.8875
c

c  data cards
c
mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 4500 0
c
sdef  cel=d1  erg=d4  rad=d2  ext=d3  pos=-6.87  -6.87  -6.505   axs=0 0 1
c
si1   l   3:1            $ path to cell 1
sp1   d   1              $ choose above with prob 1
si2   h   0.0 5.748      $ radial limits (from pos)
sp2   -21 1              $ p(r) = const*r
si3   h  -5.3825 5.3825  $ axial limits (from pos)
sp3   -21 0              $ p(z) = const
sp4   -3                 $ watt fission spectrum
```

```
c
prdmp  j  j  1  j       $ write mctal file
c
print                   $ full output
. . . . . .
e5mt.10:  converted from keno file k.10; continuos energy;  endf/b-5
c    8 bare cylinders of U-metal
c    to demonstrate restart
c
c    cell cards
c
1   1   4.80368e-2  -7  -8  9   imp:n=1  u=-1
2   0        #1              imp:n=1  u=1
3   0         -1  2  -3  4  -5  6   imp:n=1  u=2  lat=1
    fill=0:1  0:1  0:1  1  1  1  1  1  1  1  1
4   0    -11 12 -13 14 -15 16    imp:n=1  fill=2
5   0     #4  imp:n=0
c

c   surface cards
c
c  parallelpiped
1   px 0.0
2   px -13.74
3   py  0.0
4   py  -13.74
5   pz  0.0
6   pz  -13.01
c  cylinder
7   c/z  -6.87 -6.87  5.748
8   pz  -1.1225
9   pz  -11.8875
c  parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   px 13.7399
12   px -13.7399
13   py 13.7399
14   py -13.7399
15   pz 13.0099
16   pz -13.0099
c

c  data cards
c
mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
```

```
c   S(alpha,beta):  not applicable
c
c   default nergy bins;  Hansen-Roach structure
e0  1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
    1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n  1        $ ave flux in cell 1
c
c    criticality cards
kcode  300  1.0  3  103 4500 0
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=-6.87  -6.87  -6.505   axs=0 0 1
c
si1   l  4:3(1 1 0):1    $ path:  /cell4/cell3/lattice(1,1,0)/cell1
         4:3(1 0 0):1    $ etc.
         4:3(0 1 0):1    $ this ordering chosen to match
         4:3(0 0 0):1    $  sampling  in e5ce.2
         4:3(1 1 1):1    $
         4:3(1 0 1):1    $
         4:3(0 1 1):1    $
         4:3(0 0 1):1    $
sp1   1 1 1 1 1 1 1 1    $ equal probability for all paths above
c
sp2  -3                $ Watt fission spectrum
c
si3    0.0    5.748      $ radial distribution
sp3  -21  1             $ p(x) = const*abs(x)
c
si4   -5.3825  5.3825    $ axial distribution
sp4   -21  0            $ p(x)= const
c
prdmp  5  5  1  j        $ write mctal file
c
print                  $ full output
. . . . . .
message:  c 11 runtpe=inp10r

continue
. . . . . .
e5mt.12:  converted from keno file k.12; continuos energy;  endf/b-5
c
c    cell cards
c
1    1   4.8033e-2  -7  -8  9    imp:n=1
2    2   0.0980642  -17 -18 19    imp:n=1 u=1
3    3   0.106657    17: 18:-19   imp:n=1 u=1
4    0              -27 -28 29    imp:n=1     fill=1
c
11   like 1 but   trcl=(0.00  13.18  0.00)
14   like 4 but   trcl=(0.00  21.75  0.00)
```

```
c
21   like 1 but   trcl=(0.00  0.00  12.45)
24   like 4 but   trcl=(0.00  0.00  20.48)
c
31   like 1 but   trcl=(0.00  13.18 12.45)
34   like 4 but   trcl=(0.00  21.75 20.48)
c
40   0 -30 #1 #4 #11 #14 #21 #24 #31 #34  imp:n=1   $btwn cylinder & sphere
c
50   0  30                          imp:n=0   $outside of sphere

c    surface cards
c
c   finite cylinders
7    c/z  -6.59 -6.59  5.748
8    pz  -0.8425                    $ 5.3825 above midplane
9    pz  -11.6075                   $ 5.3825 below midplane
c
17   c/z  10.875 -10.875 9.525
18   pz   -1.35                     $ 8.89 above midplane
19   pz   -19.13                    $ 8.89 below midplane
c
27   c/z  10.875 -10.875 10.16
28   pz   -0.715                    $ 9.525 above midplane
29   pz   -19.765                   $ 9.525 below midplane
c
30   so   35.0                      $ enclosing sphere

c  data cards
c
mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  4.47971e-02    $ U-235
     92238.50c  2.65767e-03    $ U-238
     92234.50c  4.82716e-04    $ U-234
     92236.50c  9.57231e-05    $ U-236
c uranyl nitrate
m2     1001.50c  0.592466
       7014.50c  0.020143
       8016.50c  0.376557
      92235.50c  0.010041
      92238.50c  0.000792
c plexiglass
m3     6000.50c  0.333330
       1001.50c  0.533336
       8016.50c  0.133334
c
c    S(alpha,beta)
mt2   lwtr.01t
c
```

```
c   default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1          $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3 103 4500 0
c
sdef  cel=d1  erg=d2  rad=fcel d3  ext=fcel d4
      pos=fcel d5 axs=0 0 1
c
si1   l 1 11 21 31              $ cells
       4:2  14:2  24:2  34:2    $ path to cell 2
sp1   v                        $ prob proportional to volume
c
sp2  -3                        $ Watt fission spectrum
c
ds3   s  31 31 31 31             $ radial distrib numbers based on cell
        32 32 32 32             $ correspond to si1 card
si31    0.0  5.748             $ radial limits: source in cell 1
sp31  -21  1                   $ p(x) = const*abs(x)
si32    0.0 9.525              $ radial limits: source in cell 2
sp32  -21  1                   $ p(x) = const*abs(x)
c
ds4   s  41 41 41 41             $ radial distrib numbers based on cell
        42 42 42 42             $ correspond to si1 card
si41    -5.3825 5.3825           $ axial limits: source in cell 1
sp41  -21  0                   $ p(x) = const
si42    -8.89 8.89             $ axial limits: source in cell 2
sp42  -21  0                   $ p(x) = const
c
ds5    l -6.59 -6.59 -6.225     $ pos variable based on cell
        -6.59  6.59 -6.225     $ correspond to si1 card
        -6.59 -6.59  6.225     $
        -6.59  6.59  6.225     $
        10.875 -10.875 -10.24  $ when path is given (see si1)
        10.875 -10.875 -10.24  $ the position of the
        10.875 -10.875 -10.24  $ untranslated cell is give, so
        10.875 -10.875 -10.24  $ pos is the same each time
c
prdmp  j  j  1  j        $ write mctal file
c
print                   $ full output
. . . . . .
e5mt.13:  converted from keno file k.13; continuos energy;  endf/b-5
c   2   offset cubes of enriched U-235 surrounded by a cylindrical
c    annulus of enriched U-235
c
c   cell cards
```

```
c
1   1 4.80368e-2  -1 2 -3 4 -5 6  imp:n=1   trcl=(-0.2566 -6.35 0.0)
2   1 4.80368e-2  -1 2 -3 4 -7 6  imp:n=1   trcl=(-12.4434 -6.35 7.62)
3   1 4.80368e-2  -12 -13 6 11    imp:n=1   $ annulus
4   0             -11 -13 6 #1 #2  imp:n=1  $ btwn boxes and cylinders
5   0           (12:  13:-6)  #2   imp:n=0  $ outside
c
c   surface cards
c
c   planes

1   px  12.7
2   px  0.0
3   py  12.7
4   py  0.0
5   pz  7.62
6   pz   0.0
7   pz   11.176
c    cylinders
11   cz  13.97
12   cz  19.05
13   pz  16.18
c
c    data cards
c

mode n      $   transport neutrons only
c
c     material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 400 0
ksrc  6.35  0.0  3.81  -6.35  0.0  13.2   $  point in each block
c
prdmp  j  j  1  j             $  write mctal file
c
print
```

```
e5mt.14:  converted from keno file k.14; continuos energy;  endf/b-5
c
c  cell cards
c
1  1  4.80362e-2  -1  -8  9    imp:n=1
2  0          1  -8  9  -2   imp:n=1
3  1  4.80362e-2  -3  -8  9 2  imp:n=1
4  0      3:  8:-9     imp:n=0
c
c  surface cards
c  cylinder

1  c/z  5.08  0.0  8.89
2  cz   13.97
3  cz   19.05
c  planes
8  pz  10.109
9  pz  0.0
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1        $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 400 0
c
sdef  cel=d1  pos=fcel d2 ext=fcel d3  rad=fcel d4  erg=d5   axs=0 0 1
c
si1  l  1   3                    $ cells
sp1   v                         $prob proportional to volume
c
ds2   t  1  5.08  0.0  0.0       $  pos for cel=1
        3   0.0  0.0  0.0       $  pos for cel=2
c
ds3    s  31  32         $ ext ditrib numbers based on cell
```

```
si31   0.0  10.109     $ axial range about pos
sp31   -21  0         $  p(z)=const
si32   0.0  10.109     $ axial range about pos
sp32   -21  0         $  p(z)=const
c
ds4    s  41  42       $ rad ditrib numbers based on cell
si41   0.0  8.89      $ radial range about pos
sp41   -21  1         $  p(z)=const*abs(r)
si42   13.97 19.05     $ radial range about pos
sp42   -21  1         $  p(z)=const*abs(r)
c
sp5   -3
c
prdmp  j  j  1  j      $ write mctal file
c
print                 $ full output
. . . . . .
e5mt.15:  converted from keno file k.15; continuos energy;  endf/b-5
c
c     cell cards
c
1    1   0.04817212 -5              imp:n=1
2    2   0.106657    1 -2  3 -4      imp:n=1
3    3   0.100113    6 -7 -8 #1 #2    imp:n=1
4    0              #3 #2 #1         imp:n=0

c    surface cards
c
1    pz  -7.092175
2    pz  -4.552185
3    cz   4.1275
4    cz  12.7
5    sz   0.538475 6.5537
6    pz  -22.092175
7    pz   22.092175
8    cz   32.97
c

c  data cards
c
mode n       $ transport neutrons only
kcode  300 1.0 3 103 4500 0
sdef  cel=1 erg=d1 rad=d2 pos=0.0 0.0 0.538475
c
sp1  -3
c
si2   0.0 6.56
sp2  -21 2
c
c    material cards;  endf/b-5 data
m1   92234.50c  0.01177258     $ U-234
```

```
     92235.50c  0.97656128     $ U-235
     92236.50c  0.0019912319   $ U-236
     92238.50c  0.009674906    $ U-238
c plexiglass
m2    6000.50c  0.3333302
      1001.50c  0.5333358
      8016.50c  0.13333396
c water
m3    1001.50c  0.666667
      8016.50c  0.333333
c
c    S(alpha,beta)
mt3   lwtr.01t
c
prdmp  j  j  1  j        $ write mctal file
c
print                    $ full output
. . . . . .
e5mt.16:  converted from keno file k.16; continuos energy;  endf/b-5
c     uo2f2 infinite slab
c
c     cell cards
c
1     1 0.09872456  (-1 2 -3 4 -5 6)            imp:n=1
2     2 0.07044000  (-11 12 -3 4 -5 6)( 1: -2)  imp:n=1
3     3 0.1         (-21 22 -3 4 -5 6)(11: -12) imp:n=1
4     0             (21:-22: 3:-4: 5:-6)        imp:n=0

c    surface cards
c
c    planes
1     px  2.479
2     px -2.479
*3    py  100.0
*4    py -100.0
*5    pz  100.0
*6    pz -100.0
c
11    px  3.749
12    px -3.749
c
*21   px  17.479
*22   px -17.479

c  data cards
c
mode n        $ transport neutrons only
c
c    material cards;  endf/b-5 data
c uo2f2 solution
m1   92235.50c  0.013999    $ U-235
```

```
     92238.50c  0.001008    $ U-238
      9019.50c  0.030013
      8016.50c  0.338315
      1001.50c  0.616665
c pyrex
m2   11023.50c 2.39502e-03
     13027.50c 4.97719e-04
     14000.50c 1.80267e-02
      5010.50c 9.08241e-04
      5011.56c 3.68719e-03
      8016.50c 4.49173e-02
c borated uo2f2 solution
m3    9019.50c 2.96286e-03
      1001.50c 6.04824e-02
      8016.50c 3.32041e-02
     92235.50c 1.38188e-03
     92238.50c 9.95503e-05
      5010.50c 2.92803e-04
      5011.56c 1.18870e-03
c
c    S(alpha,beta)
mt1   lwtr.01t
mt3   lwtr.01t
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1          $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3 103 4500 0
c
ksrc    0.0  0.0  0.0          $ point in material 1
      -10.61 0.0  0.0          $ point in material 3
       10.61 0.0  0.0          $ point in material 3
c
prdmp  j  j  1  j        $ write mctal file
c
print                    $ full output
.................. ..................................................................
e5mt.17:  converted from keno file k.17; continuos energy;  endf/b-5
c
c     cell cards
c
1     1 0.0995739  -1  imp:n=1
2     0            1  imp:n=0

c    surface cards
c
```

```
1   so  16.0

c  data cards
c
mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
c uo2f2 solution
m1   92235.50c  0.0032197    $ U-235
    92238.50c  0.0002349    $ U-238
     9019.50c  0.0069084
     8016.50c  0.3344853
     1001.50c  0.6551517
c
c    S(alpha,beta)
mt1   lwtr.01t
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
    1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1        $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3 103 4500 0
c
c uniform source
sdef  cel=1  erg=d1  rad=d2 pos=0.0 0.0 0.0
c
sp1  -3                    $ Watt fission spectrum
c
si2     0.0 16.0            $ radial limits
sp2  -21  2                 $ p(x) = const*x2
c
prdmp  j  j  1  j           $ write mctal file
c
print                      $ full output
. . . . . .
e5mt.18:  converted from keno file k.18; continuos energy;  endf/b-5
c
c     cell cards
c
1    1 0.0981986   1 -2 -3            imp:n=1 u=1
2    0             1 -4 -3 #1         imp:n=1 u=1
3    2 0.106657    5 -6 -7 #1 #2       imp:n=1 u=1
4    4 0.100133-9 -5:6:7              imp:n=1 u=1
5    0           14 -15 16 -17 18 -19  imp:n=1 u=3 lat=1
      fill=-1:1 -1:1 -1:1
      1 1 1  1 1 1  1 1 1
      1 1 1  1 1 1  1 1 1
```

```
     1 1 1  1 1 1  1 1 1
6    0          20 -21 22 -23 24 -25                    imp:n=1 fill=3
7    3 0.122268   30 -31 32 -33 34 -35 (-20:21:-22:23:-24:25) imp:n=1
8    5 0.100113   30 -31 32 -33 44 -34                     imp:n=1
9    0          (-30:31:-32:33:-44:35)                  imp:n=0

c    surface cards
c
c    uranyl-nitrate cylinder
1    pz  -8.7804
2    pz   8.7804
3    cz   9.52
c
c    void cylinder-used to account for the fact that the plexiglass
c                cylinder is not completely full.
4    pz  8.9896
c
c    plexiglass cylinder
5    pz  -9.4204
6    pz   9.6296
7    cz  10.16
c
c    water filled cuboide that contains the cylinders
14   px  -18.45
15   px   18.45
16   py  -18.45
17   py   18.45
18   pz  -17.6854
19   pz   17.8946
c
c    cuboid that contains all the other cuboids
20   px  -55.3499
21   px   55.3499
22   py  -55.3499
23   py   55.3499
24   pz  -52.9199
25   pz   52.9199
c
c    cuboid that contains paraffin
30   px  -70.591
31   px   70.591
32   py  -70.591
33   py   70.591
34   pz  -68.6100999
35   pz   68.6100999
c
c    slab of water - on the negative z face
44   pz  -99.09
c

c  data cards
```

```
c
mode n      $ transport neutrons only
c
c   material #1  - aqueous uranyl nitrate
c   material #2  - plexiglass
c   material #3  - paraffin
c   material #4  - water, very low density (water vapor)
c   material #5  - water, normal density
c
c   material cards;  endf/b-5 data
c
m1    1001.50c 5.77964e-02    $ h
      7014.50c 2.13092e-03    $ n
      8016.50c 3.74130e-02    $ o
     92234.50c 1.06784e-05    $ u-234
     92235.50c 9.84599e-04    $ u-235
     92236.50c 5.29385e-06    $ u-236
     92238.50c 6.19413e-05    $ u-238
mt1   lwtr.01t
c
m2    1001.50c 0.5333358      $ h
      6012.50c 0.3333302      $ c
      8016.50c 0.13333396     $ o
c
m3    1001.50c 0.67532797     $ h
      6012.50c 0.32467203     $ c
mt3   poly.01t
c
m4    1001.50c 0.66666667     $ h
      8016.50c 0.33333333     $ o
mt4   lwtr.01t
c
m5    1001.50c 0.66666667     $ h
      8016.50c 0.33333333     $ o
mt5   lwtr.01t
c
kcode  300 1.0 3 103 4500 0
sdef  cel=d1  erg=d2  rad=d3  ext=d4 pos=0.0 0.0 0.0 axs=0 0 1
c
si1   l 6:5( 1  1  1):1  6:5(-1 -1  0):1  6:5(-1 -1  1):1
        6:5(-1  0 -1):1  6:5(-1  0  0):1  6:5(-1  0  1):1
        6:5(-1  1 -1):1  6:5(-1  1  0):1  6:5(-1  1  1):1
        6:5( 0 -1 -1):1  6:5( 0 -1  0):1  6:5( 0 -1  1):1
        6:5( 0  0 -1):1  6:5( 0  0  0):1  6:5( 0  0  1):1
        6:5( 0  1 -1):1  6:5( 0  1  0):1  6:5( 0  1  1):1
        6:5( 1 -1 -1):1  6:5( 1 -1  0):1  6:5( 1 -1  1):1
        6:5( 1  0 -1):1  6:5( 1  0  0):1  6:5( 1  0  1):1
        6:5( 1  1 -1):1  6:5( 1  1  0):1  6:5( 1  1  1):1
sp1     1 1 1  1 1 1  1 1 1    $ equal probability for all paths above
        1 1 1  1 1 1  1 1 1
        1 1 1  1 1 1  1 1 1
```

```
c
sp2  -3                    $ Watt fission spectrum
c
si3   0.0  9.52            $ radial distribution
sp3  -21  1                $ p(x) = const*abs(x)
c
si4  -8.7804 8.7804          $ axial distribution
sp4  -21  0                $ p(x) = const
c
prdmp  j  j  1  j       $ write mctal file
c
print                  $ full output
. . . . . .
e5mt.19:  converted from keno file k.19; continuos energy;  endf/b-5
c
c    cell cards
c
1   1   4.8033e-2  -7  -8  9    imp:n=1
2   2   0.0980642  -17 -18 19    imp:n=1 u=1
3   3   0.106657    17: 18:-19   imp:n=1 u=1
4   0             -27 -28 29    imp:n=1     fill=1
c
11   like 1 but   trcl=(0.00  13.18  0.00)
14   like 4 but   trcl=(0.00  21.75  0.00)
c
21   like 1 but   trcl=(0.00  0.00  12.45)
24   like 4 but   trcl=(0.00  0.00  20.48)
c
31   like 1 but   trcl=(0.00  13.18 12.45)
34   like 4 but   trcl=(0.00  21.75 20.48)
c
40   0 -30 #1 #4 #11 #14 #21 #24 #31 #34  imp:n=1   $btwn cylinder & sphere
c
50   0  30                          imp:n=0   $outside of sphere

c    surface cards
c
c   finite cylinders
7    c/z  -6.59 -6.59  5.748
8    pz  -0.8425                  $ 5.3825 above midplane
9    pz  -11.6075                 $ 5.3825 below midplane
c
17   c/z  10.875 -10.875 9.525
18   pz   -1.35                   $ 8.89 above midplane
19   pz   -19.13                  $ 8.89 below midplane
c
27   c/z  10.875 -10.875 10.16
28   pz   -0.715                  $ 9.525 above midplane
29   pz   -19.765                 $ 9.525 below midplane
c
30   so   35.0                    $ enclosing sphere
```

```
c  data cards
c
mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  4.47971e-02    $ U-235
     92238.50c  2.65767e-03    $ U-238
     92234.50c  4.82716e-04    $ U-234
     92236.50c  9.57231e-05    $ U-236
c uranyl nitrate
m2    1001.50c  0.592466
      7014.50c  0.020143
      8016.50c  0.376557
     92235.50c  0.010041
     92238.50c  0.000792
c plexiglass
m3    6000.50c  0.333330
      1001.50c  0.533336
      8016.50c  0.133334
c
c    S(alpha,beta)
mt2   lwtr.01t
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3 103 4500 0
c
sdef  cel=d1  erg=d2  rad=fcel d3  ext=fcel d4
      pos=fcel d5 axs=0 0 1
c
si1   l 1 11 21 31              $ cells
       4:2  14:2  24:2  34:2    $ path to cell 2
sp1   v                        $ prob proportional to volume
c
sp2  -3                        $ Watt fission spectrum
c
ds3   s  31 31 31 31            $ radial distrib numbers based on cell
        32 32 32 32            $ correspond to si1 card
si31    0.0  5.748             $ radial limits: source in cell 1
sp31  -21  1                   $ p(x) = const*abs(x)
si32    0.0 9.525              $ radial limits: source in cell 2
sp32  -21  1                   $ p(x) = const*abs(x)
c
ds4   s  41 41 41 41            $ radial distrib numbers based on cell
```

```
        42 42 42 42           $ correspond to si1 card
si41    -5.3825 5.3825         $ axial limits: source in cell 1
sp41  -21  0                  $ p(x) = const
si42    -8.89 8.89            $ axial limits: source in cell 2
sp42  -21  0                  $ p(x) = const
c
ds5    1 -6.59 -6.59 -6.225    $ pos variable based on cell
        -6.59  6.59 -6.225     $ correspond to si1 card
        -6.59 -6.59  6.225     $
        -6.59  6.59  6.225     $
        10.875 -10.875 -10.24  $ when path is given (see si1)
        10.875 -10.875 -10.24  $ the position of the
        10.875 -10.875 -10.24  $ untranslated cell is give, so
        10.875 -10.875 -10.24  $ pos is the same each time
c
prdmp  j  j  1  j       $ write mctal file
c
print                   $ full output
. . . . . .
e5mt.20:  converted from keno file k.20; continuos energy;  endf/b-5
c
c     cell cards
c
c     aluminum with uranyl nitrate
1    1   0.0982616  -17  -18 19   imp:n=1 u=1
2    2   0.060242    17:  18:-19  imp:n=1 u=1
3    0              -27 -28 29    imp:n=1     fill=1
c     make a triangular array of the above cans
13   like 3 but   trcl=( 21.006   0.000  0.000)
23   like 3 but   trcl=(-21.006   0.000  0.000)
33   like 3 but   trcl=( 10.503  18.192  0.000)
43   like 3 but   trcl=(-10.503  18.192  0.000)
53   like 3 but   trcl=( 10.503 -18.192  0.000)
63   like 3 but   trcl=(-10.503 -18.192  0.000)
c     enclose the array in a box
101  0    -1 2 -3  4 -5  6 #3 #13 #23 #33 #43 #53 #63  imp:n=1  $btwn
102  0    1:-2: 3:-4: 5:-6                             imp:n=0  $outside

c    surface cards
c
1    px   50.0
2    px  -50.0
3    py   50.0
4    py  -50.0
5    pz   50.0
6    pz  -0.152
c   finite cylinders
17   cz   10.16
18   pz   18.288
19   pz    0.0
c
```

```
27   cz   10.312
28   pz   18.288
29   pz   -0.152

c  data cards
c
mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.014017    $ U-235
     92238.50c  0.001010    $ U-238
      8016.50c  0.338342    $ o
      9019.50c  0.030053    $ f
      1001.50c  0.616578    $ h
m2   13027.50c  1.0         $ al can
c
c    S(alpha,beta)
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3 103 4500 0
c
c    uniformly distributed volume source in each cylinder.
c    you have to set up distributions from which to choose:
c    cell, energy, radius (from axs), and z displacement (from pos).
c    since the cylinder is in a repeated structure, but always
c    has the same cell number, you must specify the path of cells
c    which uniquely defines the cylinder you want.  the path begins
c    with the outermost cell and works down.
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=0.0 0.0 9.144  axs=0 0 1
c
si1   1  3:1                $ path:  /cell13/cell1
        13:1                $ etc.
        23:1                $
        33:1                $
        43:1                $
        53:1                $
        63:1                $
sp1        1 1 1 1 1 1 1      $ equal prob for all the above
c
sp2  -3                     $ Watt fission spectrum
c
si3    0.0  10.16            $ radial limits
sp3  -21  1                 $ p(x) = const*abs(x)
```

```
c
si4   -9.144  9.144          $ axial limits
sp4  -21  0                  $ p(x) = const
c
prdmp  j  j  1  j       $ write mctal file
c
print                  $ full output
. . . . . .
e5mt.21:  converted from keno file k.21; continuos energy;  endf/b-5
c    parially filled sphere.
c
c   cell cards
c
1    1  0.098603     -1  -3     imp:n=1 $ partially filled sphere
2    0              -1   3     imp:n=1 $ empty part of sphere
3    2  0.0602374    -2   1     imp:n=1 $ spherical tank
4    0               2         imp:n=0 $ outside of tank
c
c   surface cards

1    so  34.6
2    so  34.759
3    pz  30.0
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92234.50c  2.54223e-7 $ 2.57773e-6  $ U-234
     92235.50c  6.18922e-5 $ 6.26580e-4  $ U-235
     92238.50c  1.18834e-3 $ 1.206203e-2 $ U-238
      8016.50c  3.32845e-2 $ 3.37540e-1  $ o
      9019.50c  2.50098e-3 $ 2.53196e-2  $ f
      1001.50c  6.1567e-2  $ 6.24449e-1  $ h
m2   13027.50c  1.0         $ al
c
c    S(alpha,beta): not applicable
mt1  lwtr.01t
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 4500 0
c
sdef  cel=1  erg=d1 rad=d2 pos=0.0 0.0 0.0
```

```
c
sp1   -3                        $ watt fission spectrum
c
si2        0.0 34.6              $ radial limits
sp2   -21   2                    $ p(x) = xonst*x2
c
prdmp  j  j  1  j        $ write mctal file
c
print                   $ full output
. . . . . .
e5mt.22:  converted from keno file k.22; continuos energy;  endf/b-5
c
c    cell cards
c    This is not the optimal way to set this up; see problem 1 for
c    a better method.   This mimcs keno
1    1   4.80368e-2  -37  -38  39   imp:n=1 u=1
2    1   4.80368e-2  #1            imp:n=1 u=1
3    1   4.80368e-2  -27  -28  29   imp:n=1 u=2 fill=1
4    1   4.80368e-2  #3            imp:n=1 u=2
5    1   4.80368e-2  -17  -18  19   imp:n=1 u=3 fill=2
6    1   4.80368e-2  #5            imp:n=1 u=3
7    1   4.80368e-2  -7    -8   9   imp:n=1 u=4 fill=3
8    0              #7            imp:n=1 u=4
9    0        -1  2  -3  4  -5  6   imp:n=1 u=5 lat=1
     fill=0:1  0:1  0:1  4  4  4  4  4  4  4  4
10   0      -11 12 -13 14 -15 16    imp:n=1 fill=5
11   0              #10           imp:n=0
c
c    surface cards
c
c   parallelpiped

1    px 0.0
2    px -13.74
3    py  0.0
4    py  -13.74
5    pz  0.0
6    pz  -13.01
c   cylinder
7    c/z  -6.87 -6.87  5.748
8    pz  -1.1225              $5.3825 above midplane
9    pz  -11.8875             $5.3825 above midplane
c   parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   px 13.7399
12   px -13.7399
13   py 13.7399
14   py -13.7399
15   pz 13.0099
16   pz -13.0099
c   cylinder
17    c/z  -6.87 -6.87  5.2224
```

```
18    pz  -1.6147              $4.8903 above midplane
19    pz  -11.3953             $4.8903 above midplane
c  cylinder
27    c/z  -6.87 -6.87  4.5622
28    pz  -2.2329              $4.2721 above midplane
29    pz  -10.7771             $4.2721 above midplane
c  cylinder
37    c/z  -6.87 -6.87  3.621
38    pz  -3.1143              $3.3907 above midplane
39    pz  -9.8957              $3.3907 above midplane
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 400 0
c
c    don't bother with uniform volume source
ksrc  6.87  6.87  6.505 -6.87  6.87  6.505  $1  point per cylinder
      6.87 -6.87  6.505 -6.87 -6.87  6.505
      6.87  6.87 -6.505 -6.87  6.87 -6.505
      6.87 -6.87 -6.505 -6.87 -6.87 -6.505
c
prdmp  j j 1 j        $ write mctal file
c
print                 $ full output
. . . . . .
e5mt.23:  converted from keno file k.23; continuos energy;  endf/b-5
c     same geometry as problem 1
c     8 bare cylinders of U-metal
c     cell cards
c
1    1   4.80368e-2  -7  -8  9   imp:n=1  u=-1
2    0         #1               imp:n=1  u=1
```

```
3   0          -1  2  -3  4  -5  6   imp:n=1  u=2  lat=1
    fill=0:1  0:1  0:1  1  1  1  1  1  1  1  1
4   0     -11 12 -13 14 -15 16    imp:n=1  fill=2
5   0      #4  imp:n=0
c
c   surface cards
c
c  parallelpiped

1   px 0.0
2   px -13.74
3   py  0.0
4   py  -13.74
5   pz  0.0
6   pz  -13.01
c  cylinder
7   c/z  -6.87 -6.87  5.748
8   pz  -1.1225
9   pz  -11.8875
c  parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   px 13.7399
12   px -13.7399
13   py 13.7399
14   py -13.7399
15   pz 13.0099
16   pz -13.0099
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 400 0
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=-6.87  -6.87  -6.505   axs=0 0 1
```

```
c
si1  1  4:3(1 1 0):1   $ path:  /cell4/cell3/lattice(1,1,0)/cell1
        4:3(1 0 0):1   $ etc.
        4:3(0 1 0):1   $ this ordering chosen to match
        4:3(0 0 0):1   $  sampling  in e5ce.2
        4:3(1 1 1):1   $
        4:3(1 0 1):1   $
        4:3(0 1 1):1   $
        4:3(0 0 1):1   $
sp1   1 1 1 1 1 1 1 1   $ equal probability for all paths above
c
sp2  -3                $ Watt fission spectrum
c
si3   0.0    5.748      $ radial distribution
sp3  -21  1            $ p(x) = const*abs(x)
c
si4   -5.3825  5.3825    $ axial distribution
sp4   -21  0           $ p(x)= const
c
prdmp  j  j  1  j       $ write mctal file
c
print                  $ full output
. . . . . .
e5mt.24:  converted from keno file k.24; continuos energy;  endf/b-5
c     same geometry as problem 1, but oriented along x
c     8 bare cylinders of U-metal
c     cell cards
c
1    1   4.80368e-2  -7  -8  9   imp:n=1  u=-1
2    0         #1              imp:n=1  u=1
3    0          -1  2  -3  4  -5  6   imp:n=1  u=2  lat=1
     fill=0:1  0:1  0:1  1  1  1  1  1  1  1  1
4    0     -11 12 -13 14 -15 16    imp:n=1  fill=2
5    0      #4  imp:n=0
c
c    surface cards
c
c   parallelpiped

1    pz 0.0
2    pz -13.74
3    py  0.0
4    py  -13.74
5    px  0.0
6    px  -13.01
c   cylinder
7    c/x  -6.87 -6.87  5.748
8    px  -1.1225
9    px  -11.8875
c   parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   pz 13.7399
```

```
12   pz -13.7399
13   py 13.7399
14   py -13.7399
15   px 13.0099
16   px -13.0099
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1         $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 400 0
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=-6.87  -6.87  -6.505   axs=0 0 1
c
si1   l  4:3(1 1 0):1    $ path:  /cell4/cell3/lattice(1,1,0)/cell1
         4:3(1 0 0):1    $ etc.
         4:3(0 1 0):1    $ this ordering chosen to match
         4:3(0 0 0):1    $  sampling  in e5ce.2
         4:3(1 1 1):1    $
         4:3(1 0 1):1    $
         4:3(0 1 1):1    $
         4:3(0 0 1):1    $
sp1    1 1 1 1 1 1 1 1    $ equal probability for all paths above
c
sp2  -3                 $ Watt fission spectrum
c
si3    0.0    5.748      $ radial distribution
sp3  -21  1              $ p(x) = const*abs(x)
c
si4   -5.3825  5.3825    $ axial distribution
sp4   -21  0             $ p(x)= const
c
prdmp  j  j  1  j        $ write mctal file
c
```

```
print                   $ full output
. . . . . .
e5mt.25:  converted from keno file k.25; continuos energy;  endf/b-5
c     same geometry as problem 1, but oriented along y
c     8 bare cylinders of U-metal
c     cell cards
c
1    1   4.80368e-2  -7  -8  9   imp:n=1  u=-1
2    0         #1              imp:n=1  u=1
3    0          -1  2  -3  4  -5  6   imp:n=1  u=2  lat=1
     fill=0:1  0:1  0:1  1  1  1  1  1  1  1  1
4    0     -11 12 -13 14 -15 16    imp:n=1  fill=2
5    0      #4  imp:n=0
c
c    surface cards
c
c   parallelpiped

1    px 0.0
2    px -13.74
3    pz  0.0
4    pz  -13.74
5    py  0.0
6    py  -13.01
c   cylinder
7    c/y  -6.87 -6.87  5.748
8    py  -1.1225
9    py  -11.8875
c   parallelpiped (shrink dimensions slightly to avoid fill trouble)
11   px 13.7399
12   px -13.7399
13   pz 13.7399
14   pz -13.7399
15   py 13.0099
16   py -13.0099
c
c  data cards
c

mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1   92235.50c  0.932631    $ U-235
     92238.50c  0.055328    $ U-238
     92234.50c  0.010049    $ U-234
     92236.50c  0.001992    $ U-236
c
c    S(alpha,beta):  not applicable
c
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
```

```
    1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1          $ ave flux in cell 1
c
c     criticality cards
kcode  300  1.0  3  103 400 0
c
sdef  cel=d1  erg=d2  rad=d3  ext=d4  pos=-6.87  -6.87  -6.505   axs=0 0 1
c
si1   l  4:3(1 1 0):1    $ path:  /cell4/cell3/lattice(1,1,0)/cell1
         4:3(1 0 0):1    $ etc.
         4:3(0 1 0):1    $ this ordering chosen to match
         4:3(0 0 0):1    $  sampling  in e5ce.2
         4:3(1 1 1):1    $
         4:3(1 0 1):1    $
         4:3(0 1 1):1    $
         4:3(0 0 1):1    $
sp1   1 1 1 1 1 1 1 1    $ equal probability for all paths above
c
sp2  -3                 $ Watt fission spectrum
c
si3    0.0    5.748      $ radial distribution
sp3  -21  1              $ p(x) = const*abs(x)
c
si4   -5.3825  5.3825    $ axial distribution
sp4   -21  0             $ p(x)= const
c
prdmp  j  j  1  j        $ write mctal file
c
print                   $ full output
```

MCNP Input Listing for

MCNP Benchmark Problems

```
. . . . . .
(mcnp primer ex.1) plutonium sphere with ni coating
c     cell cards
1     1    3.8915e-2  -1     imp:n=1
2     2    9.1322e-2   1 -2 imp:n=1
3     0                2    imp:n=0

c     surface cards
1     so  6.38493
2     so  6.39763

c     data cards
mode  n
m1    94239.50c  3.7047e-2
      94240.50c  1.7510e-3
      94241.50c  1.1700e-4
m2    28000.50c  9.1322e-2
kcode  1500  1.0  3  103
ksrc    0 0 0
. . . . . .
(mcnp primer ex2.1) plutonium metal cylinder
c     cell cards
1     1    3.9802e-2  -1 2 -3    imp:n=1
2     0               1:-2:3     imp:n=0

c     surface cards
1     cz  4.935
2     pz  0
3     pz  17.273

c     data cards
mode  n
m1    94239.50c  3.9802e-2
kcode   1500  1.0  3  103
ksrc    0 0 8.6
. . . . . .
(mcnp primer ex.2.2) plutonium cylinder reflected with nat. u
c     cell cards
1     1    3.9802e-2  -1 2 -3    imp:n=1
2     2    4.7561e-2   1 -4 2 -3 imp:n=1
3     0               4:-2:3     imp:n=0

c     surface cards
1     cx  4.935
2     px  0
3     px  6.909
4     cx  9.935
```

```
c     data cards
mode  n
m1    94239.50c  3.9802e-2
m2    92238.50c  4.7219e-2
      92235.50c  3.4246e-4
kcode   1500  1.0  3  103
ksrc    3.5 0 0
. . . . . .
(mcnp primer ex.2.3) plutonium cylinder reflected with nat. u
c     cell cards
1     1    3.9802e-2  -1 2 -3     imp:n=1
2     2    4.7561e-2  -4 -6 5 #1 imp:n=1
3     0               4:-5:6     imp:n=0

c     surface cards
1     cx  4.935
2     px  0
3     px  6.909
4     cx  9.935
5     px -5.0
6     px  11.909

c     data cards
mode  n
m1    94239.50c  3.9802e-2
m2    92238.50c  4.7219e-2
      92235.50c  3.4246e-4
kcode   1500  1.0  3  103
ksrc    3.5 0 0
. . . . . .
(mcnp primer ex3) s(alpha,beta) treatment
c    parially filled cylinder.
c   cell cards
1    1  9.6586e-2    -1   3  -4            imp:n=1
2    0              -1   4  -5            imp:n=1
3    2  -2.7         (1 -2 -5 3):(-2 -3 6)  imp:n=1
4    0              2:5:-6                 imp:n=0

c   surface cards
1    cz  20.12
2    cz  20.2787
3    pz  0.0
4    pz  100.0
5    pz  110.0
6    pz  -0.1587

c  data cards
mode n       $ transport neutrons only
c
c    material cards;  endf/b-5 data
m1    1001.50c  5.7058e-2
```

```
     8016.50c  3.2929e-2
     9019.50c  4.3996e-3
    92238.50c  2.0909e-3
    92235.50c  1.0889e-4
m2   13027 1
c    default nergy bins;  Hansen-Roach structure
e0   1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
     1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
c
c  tallies
f4:n   1          $ ave flux in cell 1
c
c     criticality cards
kcode  300 1.0 3 103 4500 0
ksrc    0 0  50.0
. . . . . .
(mcnp primer ex. 4) two uo2f2 cylinders in tank of water
c     cell cards
1     1    9.9605e-2  -1 -4 5  u=3      imp:n=1  $ solution
2     0               -1  4    u=3      imp:n=1  $ void region
3     2    6.0263e-2   1:-5    u=3      imp:n=1  $ al container
4     0               -2 -3 6  fill=3   imp:n=1
5     like 4 but trcl (17 0 0)          imp:n=1
6     3    3.3428e-2   10 -11 8 -9 7 -3 #4 #5   imp:n=1
7     0               -10:11:-8:9:-7:3          imp:n=0

c     surface cards
1     cz  6.35  $ solution radius
2     cz  6.50
3     pz  80.0  $ top of container
4     pz  70.2  $ top of solution
5     pz  0.0
6     pz -0.15
7     pz -20.15 $ bottom of tank
c     sides of tank
8     px -16.5
9     px  43.5
10    py -26.5
11    py  26.5

c     data cards
mode  n
m1     92235.50c 1.1760e-3
      92238.50c 8.2051e-5
       8016.50c 3.3621e-2
       1001.50c 6.221e-2
       9019.50c 2.5161e-3
mt1   lwtr.01t
m2     13027.50c 6.0263e-2
m3      1001.50c 6.6857e-2
       8016.50c 3.3428e-2
```

```
mt3   lwtr.01t
kcode   300  1.0  3  103
ksrc    0 0 35  17 0 35
. . . . . .
(mcnp primer ex. 5) 3x2 array of pu(no3)4 cylinders
c     cell cards
1     1    9.9270e-2  -1  5 -6       u=1      imp:n=1  $ solution
2     0              -1  6 -7       u=1      imp:n=1  $ void region
3     2    8.6360e-2  -2 #1 #2       u=1      imp:n=1  $ ss container
4     0               2             u=1      imp:n=1  $ void
5     0    -8 9 -10 11 lat=1 fill=1 u=2      imp:n=1  $ lattice
6     0     13 -12 15 -14 3 -4 fill=2        imp:n=1  $ window
7     0    -13:12:-15:14:-3:4                imp:n=0  $ outside

c     surface cards
1     cz  12.49
2     cz  12.79
5     pz  0.0
6     pz  39.24
7     pz  101.7
c     beginning of lattice surfaces
8     px  17.79
9     px -17.79
10    py  17.79
11    py -17.79
c     beginning of window surfaces
3     pz -1.0
4     pz  102.7
12    px  88.949
13    px -17.789
14    py  53.369
15    py -17.789

c     data cards
mode  n
m1     94239.50c 2.7682e-4
      94240.50c 1.2214e-5
      94241.50c 8.3390e-7
      94242.50c 4.5800e-8
       1001.50c 6.0070e-2
       8016.50c 3.6540e-2
       7014.50c 2.3699e-3
mt1   lwtr.01t
m2     26000.50c 6.3310e-2
      24000.50c 1.6540e-2
      28000.50c 6.5100e-3
kcode  1500  1.0  3  103
ksrc    0 0     19.62  35.58 0     19.62  71.16 0     19.62
        0 35.58 19.62  35.58 35.58 19.62  71.16 35.58 19.62
. . . . . .
```

```
p12: jezebel at 95.5% pu-239
c     cell cards
1     1   3.9317e-2  -1  imp:n=1
2     0              1  imp:n=0

c     surface cards
1     so  6.385

c     data cards
mode  n
m1    94239.50c 3.7555e-2
      94240.50c 1.7622e-3
kcode  1500  1.0  3  103
ksrc    0. 0. 0.
. . . . . .
p13: jezebel at 80% pu-239
c     cell cards
1     1   3.9593e-2  -1  imp:n=1
2     0              1  imp:n=0

c     surface cards
1     so  6.660

c     data cards
mode  n
m1    94239.50c 3.1701e-2
      94240.50c 7.8922e-3
kcode   300  1.0  3  103
ksrc    0. 0. 0.
. . . . . .
p21: uranium cylinder with 10.9% u-235
c     cell cards
1     1   4.7196e-2  -1 2 -3   imp:n=1
2     0              -4 #1     imp:n=1
3     0               4        imp:n=0

c     surface cards
1     cy  26.65
2     py  0.
3     py  119.392
4     so  130.

c     data cards
mode  n
m1    92235.50c 5.2028e-3
      92238.50c 4.1993e-2
kcode   300  1.0  3  103
sdef  axs 0 1 0  pos 0 60 0  ext d1  rad d2
si1   55
si2   .1 26
print
```

```
. . . . . .
p22: uranium cylinder with 14.11% u-235
c    cell cards
1    1   4.6658e-2  -1 2 -3   imp:n=1
2    0              -4 #1     imp:n=1
3    0               4        imp:n=0

c    surface cards
1    cy  26.65
2    py  0.
3    py  44.239
4    so  55

c    data cards
mode  n
m1    92235.50c 6.6555e-3
      92238.50c 4.0002e-2
kcode   300  1.0  3  103
sdef  axs 0 1 0  pos 0 22 0  ext d1  rad d2
si1   22
si2   .1 26
print
. . . . . .
p31: graphite reflected uranium sphere 93.3% u-235
c    cell cards
1    1   4.8128e-2  -1    imp:n=1
2    2   8.3500e-2   1 -2 imp:n=1
3    0               2    imp:n=0

c    surface cards
1    so  7.39840
2    so  12.49840
3    so  55

c    data cards
mode  n
m1    92235.50c 4.5037e-2
      92238.50c 3.0914e-3
m2     6012.50c 8.3389e-2
      26000.50c 6.1227e-5
      16032.50c 5.0181e-5
mt2   grph.01t
kcode  1500  1.0  3  103
sdef  axs 0 1 0  pos 0 0 0  ext d1  rad d2
si1   6
si2   6
print
. . . . . .
p32: water reflected uranium sphere 97.67% u-235
c    cell cards
1    1   4.8150e-2  -1          imp:n=1
```

```
2    2   1.0019e-1   1 -2 -3 4  imp:n=1
3    0              -5 (2:3:-4) imp:n=1
4    0               5          imp:n=0

c    surface cards
1    so  6.5537
2    cy  30
3    py  35
4    py -35
5    so  50

c    data cards
mode  n
m1    92234.50c 5.3000e-4
      92235.50c 4.7030e-2
      92236.50c 1.0000e-4
      92238.50c 4.9000e-4
m2     1001.50c 6.6790e-2
       8016.50c 3.3400e-2
mt2   lwtr.01t
kcode 300  1.0  3  103
ksrc  0. 0. 0.
print
. . . . . .
p41: 3 uo2f2 cylinders in triangle configuration
c    cell cards
1    1   1.0270e-1     -1 3 -4         imp:n=1 u=-1
2    2   6.0486e-2   1:-3:4          imp:n=1 u=1
7    0              -2 8 -9 fill=1  imp:n=1
5    0              -7 #7 #8 #9     imp:n=1
6    0               7              imp:n=0
8    like 7 but trcl=1
9    like 7 but trcl=2

c    surface cards
1    cy  10.15
2    cy  10.30
3    py  0.0
8    py -0.15
9    py  41.55
4    py  41.40
7    so  150

c    data cards
mode  n
m1    92235.50c 2.1936e-4
      92238.50c 1.5736e-5
       1001.50c 6.7709e-2
       8016.50c 3.4291e-2
       9019.50c 4.6964e-4
m2    13027.50c 6.0486e-2
```

```
mt1   lwtr.01t
tr1   20.98 0 0
tr2   10.49 0 18.169
kcode 1500  1.0  3  103
sdef  axs 0 1 0  pos d1  rad d2  ext d3
sp1   0.33 0.33 0.34
si1   l 0 20.7 0  20.98 20.7 0  10.49 20.7 18.169
si2   8
si3   15
print
. . . . . .
```

MCNP Input Listing for

Additional Benchmark Problems

```
. . . . . .
prob.1 water reflected pu metal sphere
c     cell cards
1     1    4.8973e-2  -1     imp:n=1
2     2    1.0011e-1   1 -2 imp:n=1
3     0               2     imp:n=0

c     surface cards
1     so  4.11
2     so  24.11

c     data cards
mode  n
m1     94239.50c 4.8973e-2
m2      1001.50c 6.6737e-2
       8016.50c 3.3368e-2
mt2  lwtr.01t
kcode   300  1.0  3  103  300  0
sdef  cel=1
prdmp
print
. . . . . .
prob.2 water reflected pu nitrate sphere
c     cell cards
1     1    1.0111e-1  -1     imp:n=1
2     2    8.4795e-2   1 -2 imp:n=1
3     3    9.9095e-2   2 -3 imp:n=1
4     0               3     imp:n=0

c     surface cards
1     so  14.568
2     so  14.692
3     so  39.6922

c     data cards
mode  n
m1     94239.50c 3.3662e-4
      94240.50c 1.6163e-5
       7014.50c 2.7595e-3
       1001.50c 6.0260e-2
       8016.50c 3.7734e-2
m2     26000.50c 5.8886e-2
      24000.50c 1.7672e-2
      28000.50c 8.2374e-3
m3      1001.50c 6.6063e-2
       8016.50c 3.3032e-2
mt1   lwtr.01t
```

```
mt3  lwtr.01t
kcode  300  1.0  3  103
sdef  cel=1
prdmp
print
. . . . . .
prob.3 bare pu nitrate solution sphere
c     cell cards
1     1    9.9987e-2  -1     imp:n=1
2     2    6.0260e-2   1 -2 imp:n=1
3     0                2     imp:n=0

c     surface cards
1     so  61.01
2     so  61.79

c     data cards
mode  n
m1    94239.50c 2.3201e-5
      94240.50c 6.0140e-7
      94241.50c 1.7720e-8
       7014.50c 7.6043e-4
       1001.50c 6.4836e-2
       8016.50c 3.4367e-2
m2    13027.50c 6.0260e-2
mt1   lwtr.01t
kcode 300  1.0  3  103
sdef  cel=1
prdmp
print
. . . . . .
prob.4 bare u-233 nitrate solution in a sphere
c     cell cards
1     1    1.0060e-1  -1     imp:n=1
2     2    6.0263e-2   1 -2 imp:n=1
4     0                2     imp:n=0

c     surface cards
1     so  34.595
2     so  34.9252

c     data cards
mode  n
m1    92233.50c 4.3293e-5
      92234.50c 7.1479e-7
      92235.50c 1.7574e-8
      92238.50c 2.7763e-7
       8016.50c 3.3721e-2
       7014.50c 8.8300e-5
       1001.50c 6.6747e-2
m2    13027.50c 6.0263e-2
```

```
mt1  lwtr.01t
kcode  300  1.0  3  103  300
sdef  cel=1
prdmp
print
. . . . . .
prob.5 godiva
c     cell cards
1     1    4.8032e-2  -1  imp:n=1
2     0               1  imp:n=0

c     surface cards
1     so  8.72

c     data cards
mode  n
m1    92234.50c    1.005e-2
      92235.50c    9.3263e-1
      92236.50c    1.9993e-3
      92238.50c    5.532e-2
tmp 2.5851e-8
kcode   300  1.0  3  103  300  0
sdef  cel=1
prdmp
print
. . . . . .
prob.6 bare u-235 nitrate solution in a sphere
c     cell cards
1     1    1.0015e-1  -1    imp:n=1
2     2    6.0263e-2   1 -2 imp:n=1
3     0                2    imp:n=0

c     surface cards
1     so  34.595
2     so  34.9252

c     data cards
mode  n
m1    92235.50c 4.8066e-5
      92234.50c 5.3800e-7
      92236.50c 1.3800e-7
      92238.50c 2.8070e-6
       8016.50c 3.3687e-2
       7014.50c 1.8690e-4
       1001.50c 6.6228e-2
m2    13027.50c 6.0263e-2
mt1   lwtr.01t
kcode   300  1.0  3  103  300  0
sdef  cel=1
prdmp
print
```

```
. . . . . .
prob.7 low enriched unmoderated mox sphere
c     cell cards
1     1    7.1496e-2  -1     imp:n=1
2     2    1.0002e-1   1 -2 imp:n=1
3     0               2     imp:n=0

c     surface cards
1     so  19.4
2     so  39.4

c     data cards
mode  n
m1     94239.50c 1.9591e-3
      92235.50c 1.6460e-4
      92238.50c 2.2450e-2
       8016.50c 4.9148e-2
m2      1001.50c 6.6680e-2
       8016.50c 3.3343e-2
mt2   lwtr.01t
kcode   300  1.0  3  103
sdef  cel=1
prdmp
print
. . . . . .
prob.8 water reflected pu-u solution in cyl.
c     cell cards
1     1    1.0005e-1  -1 -2  3                imp:n=1
2     0               -1  2 -5                imp:n=1
3     2    8.6360e-2  (1 -4  3 -5):(-4 -3 6)   imp:n=1
4     3    1.0002e-1  (-7 4 6 -5):(-7 -6 8) #3 imp:n=1
5     0               #4                      imp:n=0

c     surface cards
1     cz  30.48
2     pz  95.20
3     pz  0.0
4     cz  30.559
5     pz  106.6
6     pz -0.95
7     cz  45.559
8     pz -20.95

c     data cards
mode  n
m1     94238.50c 8.3767e-9
      94239.50c 2.9365e-5
      94240.50c 1.7386e-6
      94241.50c 1.0752e-7
      94242.50c 2.1964e-8
      92235.50c 5.0609e-7
```

```
     92236.50c 1.4077e-8
     92238.50c 7.5163e-5
      1001.50c 6.3896e-2
      8016.50c 3.4957e-2
      7014.50c 1.0883e-3
m2    26000.50c 6.3310e-2
     24000.50c 1.6540e-2
     28000.50c 6.5100e-3
m3     1001.50c 6.6680e-2
      8016.50c 3.3343e-2
mt1  lwtr.01t
mt3  lwtr.01t
kcode  300  1.0  3  103
sdef  cel=1
prdmp
print
. . . . . .
prob.9 water reflected pu-u-gd solution in cyl.
c     cell cards
1     1    9.8242e-2  -1 -2  3              imp:n=1
2     0              -1  2 -5              imp:n=1
3     2    8.6360e-2  (-4 1 3 -5):(-4 -3 6)    imp:n=1
4     3    1.0002e-1  (-7 4 6 -5):(-7 -6 8) #3 imp:n=1
5     0              #4                    imp:n=0

c     surface cards
1     cz  30.154
2     pz  75.3
3     pz  0.0
4     cz  30.593
5     pz  106.6
6     pz -0.95
7     cz  45.593
8     pz -20.95

c     data cards
mode  n
m1    94238.50c 2.3500e-6
     94239.50c 1.8260e-4
     94240.50c 1.1180e-5
     94241.50c 7.1600e-7
     94242.50c 1.1900e-7
     92234.50c 2.7000e-8
     92235.50c 3.0130e-6
     92236.50c 5.4000e-8
     92238.50c 4.4420e-4
      1001.50c 5.5178e-2
      8016.50c 3.8688e-2
      7014.50c 3.7278e-3
     64152.50c 8.1189e-9
     64154.50c 8.8496e-8
```

```
     64155.50c 6.0080e-7
     64156.50c 8.3097e-7
     64157.50c 6.3530e-7
     64158.50c 1.0084e-6
     64160.50c 8.8739e-7
m2    26000.50c 6.3310e-2
     24000.50c 1.6540e-2
     28000.50c 6.5100e-3
m3     1001.50c 6.6680e-2
      8016.50c 3.3343e-2
mt1   lwtr.01t
mt3   lwtr.01t
kcode  300  1.0  3  103
sdef  cel=1
prdmp
print
. . . . . .
prob.11 4x4x4 array of pu metal cylinders
c     cell cards
1     1    4.9209e-2  -1 -2 3                    u=1 imp:n=1
2     0              #1                        u=1 imp:n=1
3     0             -4 5 -6 7 -8 9    lat=1 fill=1 u=2 imp:n=1
4     0              11 -10 13 -12 15 -14   fill=2     imp:n=1
5     0              #4                          imp:n=0

c     surface cards
1     cz  3.2625
2     pz  2.3165  $ top of cyl
3     pz -2.3165  $ bottom of cyl
c     unit cell around cylinder
4     px  6.255
5     px -6.255
6     py  6.255
7     py -6.255
8     pz  3.93
9     pz -3.93
c     window surfaces (shrink dimensions to avoid fill trouble)
10    px  43.784
11    px -6.254
12    py  43.784
13    py -6.254
14    pz  27.509
15    pz -3.929

c     data cards
mode  n
m1    94239.50c 4.6053e-2
     94240.50c 2.9263e-3
     94241.50c 2.2454e-4
     94242.50c 4.8612e-6
kcode  1500  1.0  3  103
```

```
sdef  pos=18.765 18.765 11.79  axs=0 0 1 rad=d1 ext=d2
si1  0.0001 20
si2  15
prdmp
print
. . . . . .
sheba2
c     cell cards
1     0     -1 -2 3     imp:n=1
2     2     8.2974e-2  (-4 +1 -2 3) : (-4 5 -3)     imp:n=1
3     1     9.3607e-2  -6 4 -7 8    imp:n=1
4     0     -6 4 7 -2                imp:n=1
5     2     8.2974e-2  (-9 -10 2) :(-9 6 -2 8) : (-9 4 -8 5) : (-9 -5 11)
      imp:n=1
6     0     9: 10: -11     imp:n=0

c     surface cards
1     cz  2.36
2     pz   36.21
3     pz  -38.75
4     cz  3.0
5     pz   -39.39
6     cz  23.9
7     pz    6.00
8     pz   -36.85
9    cz  25.4
10    pz   38.755
11    pz   -39.391

c     data cards
mode  n
m1     92238.50c     2.4990e-3
       92235.50c     1.3190e-4
       92234.50c     6.8450e-7
       92236.50c     1.3160e-6
        8016.50c     3.2100e-2
        9019.50c     5.3340e-3
        1001.50c     5.3540e-2
m2     26000.50c     5.7107e-2
       24000.50c     1.6768e-2
       28000.50c     7.4279e-3
       25055.50c     1.6706e-3
mt1  lwtr.01t
e0    1.0e-7  4.0e-7  1.0e-6  3.0e-6 1.0e-5  3.0e-5  1.0e-4 5.5e-4  3.0e-3
      1.7e-2  0.1  0.4  0.9  1.4  3.0  20.0
f4:n    3
kcode 300  1.0  3  103
sdef  cel=3  erg=d3  rad=d1  ext=d2  pos=0 0 0   axs=0 0 1
si1   3.0  23.9
si2   -36.85  0.0
sp3   -3
```

prdmp
print

Appendix F

Instructions for Executing PKENO V.a

Instructions for Executing PKENO V.a

PKENO V.a is a continuous energy version of the multigroup Monte Carlo code KENO V.a. Although extensive modifications were made to KENO in the development of PKENO, the modifications are relatively transparent to the user. In particular, a novice KENO V.a user can execute PKENO by making minor modifications to the mixing table data in the input file. The following is a brief tutorial describing the PKENO input file.

The PKENO input file expects a nuclide specified in a ZZZAAA NNC format. The ZZZAAA number is a combination of atomic number and mass for the nuclide, NN identifies the data file and C denotes continuous energy. The available cross section data for specific nuclides is listed in Refs. 4 and 36. There must be a blank space following the ZZZAAA number and file extension. With a blank space after the file extension, the nuclide's atomic number density (atoms/b-cm) and system temperature (degrees Kelvin) is specified. The mixing table specifications for a mixture i must be in the following format:

mix=i zzzaaa nnc atom density temperature

PKENO also reads a thermal scattering law, S(α,B), identifier specified in a ZAID NNT format. The ZAID identifier specifies the compound, nn identifies the data file, and T denotes thermal scattering law data. As in the nuclide specification, there must be a blank between the ZAID identifier and file extension. The S(α,β) data is provided for a specific number of compounds, which are also listed in Refs. 4 and 36, at various temperatures. The S(α,β) data card (**sab=**) is assigned a number which

corresponds to a mixture in the mixing table. The sab number identifies the mixture in which thermal scattering law data will be applied. The sab identifier must be the first data specified in the mixing table and can only apply to one mixture. If additional mixtures require the same or different scattering law data, a separate sab identifier must be supplied for each mixture. The S(α,β) specification for thermal scattering law data for mixture i must be in the following format:

sab=i zaid nnt

For example, KENO V.a test problem 21 is a partially filled sphere of UO_2F_2 solution at room temperature. The PKENO input file for this problem would have the following format:

```
sample problem 21  partially filled sphere of uo2f2 solution
read geom
hemisphe-z  1 1 34.6   chord 30.
sphere      0 1 34.6
sphere      2 1 34.759
end geom
read mixt
sab=1  lwtr  01t
mix=1  1001  50c 6.15670e-02 293
       8016  50c 3.32845e-02 293
       9019  50c 2.50098e-03 293
       92234 50c 2.54223e-07 293
       92235 50c 6.18922e-05 293
       92238 f50c 1.18834e-03 293
mix=2  13027 50c 6.02374e-02 293
end mixt
end data
```

The light water S(α,β) treatment is used because the Hydrogen atoms present in the mixture are bound in the water molecule.

In order to use the current biasing schemes in KENO without modification, it is

necessary to employ an energy group structure. In addition, collecting calculated quantities (e.g., flux, leakage, etc.) by energy groups is statistically more efficient. During the input processing, PKENO V.a reads a user defined energy group structure for the continuous energy calculation. The user must specify the group structure in a file named egs. PKENO uses the AMPX 27 group structure by default.

Currently, PKENO is only available in stand alone mode, and the executable file name is pkenova. In order to execute the code, the input file must be renamed input, and the pkenova command is typed at the system prompt.

Vita

Michael E. Dunn was born in Knoxville, Tennessee on February 22, 1970. He graduated from Central High School in June 1988. The following August he entered the University of Tennessee, Knoxville, where he received a Bachelor of Science degree in Nuclear Engineering in August 1992.

In August 1992 he accepted a Department of Energy (DOE) Nuclear Engineering & Health Physics Fellowship and began study toward a Master of Science degree in Nuclear Engineering at the University of Tennessee, Knoxville. He received the M.S. degree in August 1994. He then began study toward a Doctor of Philosophy Degree in Nuclear Engineering at the University of Tennessee. He expects to receive the Ph.D. degree in December 1996.